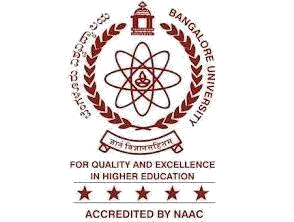
**BANGALORE UNIVERSITY**

## UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K R Circle, Bengaluru - 560001

# Department of Computer Science and Engineering A DBMS Mini Project Report on

## “GROCERY SHOP MANAGEMENT”

**Submitted By**

Siddalingaswamy V H(19GANSE043)

V SEM, B.Tech (ISE)

Yatish K V(14GANSE059)

V SEM, B.Tech (ISE)

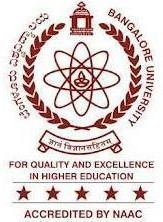
**Under the Guidance of**

|  |  |  |
| --- | --- | --- |
| **Dr. Samyama Gunjal G H**  **Assistant Professor** | **&** | **Miss. Shruti Gupta**  **Guest Faculty** |
| **Dept of CSE** |  | **Dept of CSE** |
| **UVCE** |  | **UVCE** |

**YEAR 2021-2022**

Bangalore University

University Visvesvaraya College of Engineering

K R Circle, Bengaluru - 560001

**Department of Computer Science and Engineering Certificate**

This is to certify that Mr. Yatish K V of V Semester, B.Tech (Information Science and Engineering) bearing the register number 19GANSE059 has submitted the Database Management System Mini Project Report on “GROCERY SHOP MANAGEMENT”-A Bill Generating and Stock management System in partial fulfillment for the Database Management System Lab prescribed by the Bangalore University for the academic year 2021-22.

Guide: Chairperson:

|  |  |
| --- | --- |
|  |  |
| **Dr. Samyama Gunjal G H** | **Dr. Dilip Kumar S M** |
| **Assistant Professor** | **Professor &Chairperson** |
| **Department of CSE** | **Department of CSE** |
| **UVCE** | **UVCE** |
| Examiner 1: |  |
| Examiner 2: |  |

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task is incomplete without the mention of people who made it possible, whose constant guidance and encouragement crown all efforts and success.

I am grateful to acknowledge **Dr. Venugopal K R**, Vice Chancellor, Bangalore University, for his valuable suggestions and relentless support, which has sustained me throughout the course of the project.

I express my gratefulness to **Dr. H N Ramesh**, Principal, UVCE who has been leading our college with a brighter vision in technical education.

I express my profound gratitude to **Dr. Dilip Kumar S M**, Chairperson, Department of Computer Science and Engineering, UVCE who has been a constant source of inspiration to the students work.

With extreme honour I express my deep sense of gratitude to my guide **Smt. Samyama Gunjal G H**, Assistant Professor, Department of Computer Science and Engineering, UVCE for her valuable guidance and supervision in this course of seminar.

I express my sincere thanks to all teaching and non-teaching staff, Department of Computer Science and Engineering, UVCE for all the facilities that they provided me to successfully complete this project.

I also thank my parents and friends for their continuous support and encouragement.

### THANK YOU

**SIDDALINGASWAMY V H**

**19GANSE043**

**YATISH K V**

**19GANSE059**

# ABSTRACT

Grocery Management is a Database Management System project, designed to manage the grocery shop in an efficient way, with the ease of maintaining records of employees, stock, bills and other necessary components of the grocer shop. Increased demand of grocery shoppers generated the need for much attention for the shopping industry. Providing much option with ease and speed of buying is the need of the hour. Technological interference has become mandatory to improve the quality of the service and business in this industry. Our proposed system is a stock management system that enables ease for the grocery shop owners. It overcomes the disadvantages of the traditional systems. Our proposed system is a medium to generate bills in a hassle free form for the grocery shop employees. The bill generating system sets up a grocery list from which the groceries bought by the customers and the quantity can be input and the final bill can be generated and stored. This is an application designed by using the latest tools such as Tkinter, MYSQL primarily for use in the grocery shop industry. This system will allow grocery shops to increase efficiency of business by reducing the labor cost involved. The system also allows to quickly and easily manage stock which customers can browse and buy.

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
| i | Acknowledgement | I |
| ii | Abstract | II |
| iii | Table of Contents | III |
|  |  |  |
| 1 | Introduction   * 1. Database Management System   2. Entities and Attributes   3. Relationships   4. ER Diagram   5. Structured Query Language | 1 |
| 2 | Literature Review  2.1 Analysis of Existing System  2.2 Analysis of New System  2.3 Software Used | 12 |
| 3 | Proposed Work | 20 |
|  | 3.1 ER Diagram of Grocery Shop Management |  |
|  | 3.2 Relational Database Schema  3.3 Database Relations  3.4 Normalization of Relations |  |
| 4 | Results | 29 |
|  | Conclusion and Future Work | 32 |
|  | References | 33 |
|  |  |  |
|  |  |  |

# INTRODUCTION

This project sets to develop a stock management and bill generating app for a Grocery shop. Many industries are now quickly adopting new technologies. Sales industry also embraces different types of technologies which make daily processes easier and faster.

Grocery shops can use these technologies in different ways. One of them is to build an application which may help in managing the stock and generating bills and storing them in an orderly manner. When it comes to buying groceries, customers want to be ensured about the quality of the groceries that she/he is going to buy. There are also some problems of traditional grocery management system, which results wasting time and making conflicts. In existing system there are few problems:

* It becomes very hard to keep track of the current stock after each purchase.

In this method time and manual work is required. Some grocery shops keep track via books. While this may be easy, searching for the required product in the book is very hard.

To solve these issues, a Grocery management System has been developed which is originally designed for small scale business. But this system is applicable in any grocery shop. The main advantage of this system is that it greatly simplifies the stock management and bill generating process for the shop.

The anticipated benefits of the project are:

1. This will speed up the billing process.
2. The system will help to reduce labor cost involved.
3. This will avoid long queues at the counter due to the speed of execution.
4. The system will be less probable to make mistake.
5. Since employees are not tied up counting stock, labor cost is reduced.
6. Order accuracy is another benefit for the shop.

## Database Management System (DBMS)

DBMS is a collection of programs that enables users to create and maintain a database The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

A Relational database is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes. In typical solutions, tables may have additionally defined relationships with each other. In the relational model, each table schema must identify a column or group of columns, called the primary key, to uniquely identify each row. A relationship can then be established between each row in the table and a row in another table by creating a foreign key, a column or group of columns in one table that points to the primary key of another table.

### 1.1.1 Characteristics of Database Management Systems

* Self-describing nature.
* Keeps a tight control on data redundancy.
* Enforces user defined rules to ensure that integrity of table data.
* Provides insulation between Programs and data, Data abstraction.
* Supports multiple views of the data.
* Helps sharing of data and Multi-user transaction processing.

### 1.1.2 Advantages of using the DBMS approach

* Controlling the redundancy.
* Restricting unauthorized access.
* Providing persistent storage for program objects.
* Providing storage structures for efficient query processing.
* Providing backup and recovery.
* Providing multiple users interfaces.
* Enforcing Integrity Constraints.
* Representing Complex Relationships among Data.

## 1.2 ENTITIES AND ATTRIBUTES

An entity may be an object with a physical existence (for e.g. A particular person, car or employee) or it may be an object with a conceptual existence (e.g., company, university).

Each entity has attributes i.e., the particular properties that describe it. The attribute values that describe each entity become a major part of the data store in the database. Whenever an attribute of one entity type refers to another entity type, a relationship exists. In the initial design of entity types, relationships are typically captured in the form of attributes. As the design is refined these attributes get converted into relationships between entity types. In the ER diagrams, the emphasis is on representing the schemas rather than the instances. This is more useful in the database design because a database schema changes rarely, whereas contents of the entity sets change frequently.

### 1.2.1 Types of Attributes

* + - 1. **Simple attribute: -** An attribute is classified as a simple attribute if it cannot be partitioned into smaller components. For example, age and sex of a person. A simple attribute is represented by an oval.
      2. **Composite attribute: -** A composite attribute can be subdivided into smaller components which further form attributes. For example, ‘name’ attribute of an entity “person” can be broken down into first name and last name which further form attributes. Grouping of these related attributes forms a composite attribute.
      3. **Single valued attribute: -** If an attribute of a particular entity represents single value for each instance, then it is called a single-valued attribute. For example, Ramesh, Kamal and Suraj are the instances of entity ‘student’ and each of them is issued a separate roll number. A single oval is used to represent this attribute.
      4. **Multi valued attribute: –** An attribute which can hold more than one value, it is then termed as multi-valued attribute. For example, phone number of a person. Symbol of multi- valued attribute is shown below

Phone number

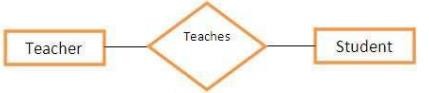
**Symbol of Multivalued attribute**

* + - 1. **Derived attribute:** A derived attribute calculates its value from another attribute. For example, ‘age’ is a derived attribute if it calculates its value from ‘current date’ & ‘birth date’ attributes. A derived attribute is represented by a dashed oval.
      2. **Stored attribute:** In some cases, two or more attribute values are related for example, the Age and Birth\_date attributes of a person. For a particular person entity, the value of Age can be determined from the current date and the value of the person’s Birth\_date. The Age attribute hence called derived attribute and is said to be derivable from the Birth\_date attribute, which is called a Stored attribute.
      3. **Complex attribute**: is a Composite and Multivalued attribute.

## 1.3 RELATIONSHIPS

A relationship is defined as bond or attachment between 2 or more entities. Normally, a verb in a sentence signifies a relationship. For example,

* An employee assigned a project.
* Teacher teaches a student.
* Author writes a book.

A diamond is used to symbolically represent a relationship in the ER diagram.

**Relationship connecting two Entities.**

### Various terms related to relationships

**1.3.1 Degree of relationship: -** It signifies the number of entities involved in a relationship. Degree of a relationship can be classified into following types:

* Unary relationship: - If only single entity is involved in a relationship then it is a unary relationship. For example, an employee (manager) supervises another employee.
* Binary relationships: - when two entities are associated to form a relation, then it is known as a binary relationship. For example, A person works in a company. Most of the times we use only binary relationship in an e-r diagram.
* Other types of relationships are ternary and quaternary. As the name signifies, a ternary relationship is associated with three entities and a quaternary relationship is associated with four entities.

**1.3.2 Connectivity of a relationship: -** Connectivity of a relationship describes, how many instances of one entity type are linked to how many instances of another entity type. Various categories of connectivity of a relationship are:

* **One to One (1:1) –** “Customer gets Bill” signifies a one-to-one relationship because only one instance of an entity is related with exactly one instance of another entity

type.

**GETS**

**1**

**1**

**BILL**

**CUSTOMER**

**One to One Relationship**

* **One to Many (1:M) –** “Employee manage Customer” is a one-to-many relationship because a Customer can place more than one order, but a order is related to only one Customer.

**1**

**M**

**CUSTOMER**

**EMPLOYEE**

**MANAGE**

**One to Many Relationship**

* **Many to One (M:1) –** “Employee has role” is a many-to-one relationship.

**HAS**

**M**

**1**

**EMPLOYEE**

**ROLE**

**Many to One Relationship**

* **Many to Many (M: N) –** “Customer buys Product” is a many-to-many relationship because an author can write many books and a book can be written by many authors.

**BUYS**

**M**

**M**

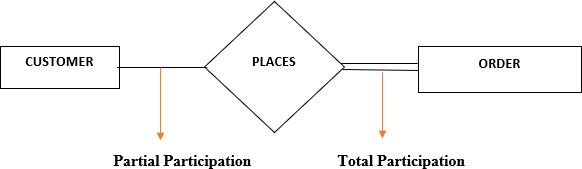
**PRODUCT**

**CUSTOMER**

**Many to Many Relationship**

### 1.3.3 Total Participation and Partial Participation

* **Total Participation:** It specifies that each entity in entity set must compulsorily participate in at least one relationship instance in that relationship set. It is also called as Mandatory Participation. The total participation is represented using a double line between the entity set and the relationship set.
* **Partial Participation:** It specifies that each entity in entity set may or may not participate in the relationship instance in that relationship set. It is also called as Optional Participation. The partial participation is represented using a single line between the entity set and the relationship set.



**Total and Partial Participations**

### Types of Entities

**Strong entity:** A strong entity has a primary key attribute which uniquely identifies each entity. Symbol of strong entity is same as an entity.



**Strong Entity**

**Weak entity:** A weak entity does not have a primary key attribute and depends on other entity via a foreign key attribute.

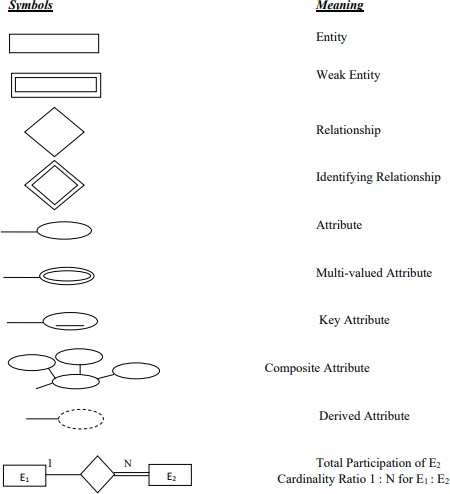
**Weak Entity**

## 1.4 ER DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship amongtables and their attributes, ER diagram shows the complete logical structure of a database. An Entity-Relationship (ER) model is an abstract way to describe a database. It is a popular high-level conceptual data model. Entity relationship diagrams (ER diagrams) are used to present the diagrammatic notations associated with ER model.

### Notations for ER diagram

****

**Notations for ER diagram**

## 1.5 Structured Query Language (SQL)

SQL is a comprehensive database language. SQL is a standard language for storing, manipulating and retrieving data in databases. The ANSI standard SQL provides basic functions for data manipulation, transaction control, and record retrieval from the database. However, most end users interact with Oracle through application that provides an interface that hides the underlying SQL and its complexity. SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively.

### 1.5.1 Applications of SQL

SQL is one of the most widely used query language over the databases that:

* Allows users to access data in the relational database management systems.
* Allows users to describe the data.
* Allows users to define the data in a database and manipulate that data.
* Allows to embed within other languages using SQL modules, libraries & pre- compilers.
* Allows users to create and drop databases and tables.
* Allows users to create view, stored procedure, functions in a database.
* Allows users to set permissions on tables, procedures and views.

### 1.5.2 SQL Commands

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature –

### DDL - Data Definition Language

|  |  |
| --- | --- |
| **SL.No.** | **Command & Description** |
| 1 | **CREATE**  Creates a new table, a view of a table, or other object in the database. |
| 2 | **ALTER**  Modifies an existing database object, such as a table. |
| 3 | **DROP**  Deletes an entire table, a view of a table or other objects in the database. |

**DML - Data Manipulation Language**

|  |  |
| --- | --- |
| **SL.No.** | **Command & Description** |
| 1 | **SELECT**  Retrieves certain records from one or more tables. |
| 2 | **INSERT**  Creates a record. |
| 3 | **UPDATE**  Modifies records. |
| 4 | **DELETE**  Deletes records. |

### DCL - Data Control Language

|  |  |
| --- | --- |
| **SL.No.** | **Command & Description** |
| 1 | **GRANT**  Gives a privilege to user. |
| 2 | **REVOKE**  Takes back privileges granted from user. |

**1.5.3 SQL Data Type**

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression have a related data type in SQL. You can use these data types while creating your tables. You can choose a data type for a table column based on your requirement.

### Numeric data types:

1. INT(size) - A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295.
2. INTEGER(size) - Equal to INT(size).
3. SMALLINT(size) - A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535.
4. FLOAT*(*size*,* d*)* - A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.
5. DOUBLE*(*size*,* d*)* - A normal-size floating point number. The total number of digits is specified in size*.* The number of digits after the decimal point is specified in the d parameter.
6. DECIMAL*(*size*,* d*)* - An exact fixed-point number. The total number of digits is specified in size*.* The number of digits after the decimal point is specified in the d parameter.

### Character-String data types:

1. CHAR(size) - A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1.
2. VARCHAR(size) - A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535.
3. CHAR VARYING(n) – similar to VARCHAR(size).

### Bit-String data types:

1. BIT(size) - A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
2. BIT VARYING(n) - similar to BIT(size) where n is the maximum number of bits. The default for n ,the length of a character string or bit string is 1.
3. BINARY(size) - Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1.

### Boolean data types:

1. BOOL - Zero is considered as false, nonzero values are considered as true.
2. BOOLEAN - Equal to BOOL.

### Date and Time data types:

1. DATE **-** A date. Format: YYYY-MM-DD. The supported range is from '1000- 01-01' to '9999-12-31'.
2. DATETIME- A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
3. TIME- A time. Format: hh:mm:ss. The supported range is from '- 838:59:59' to '838:59:59'.
4. TIMESTAMP - TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss.

### 1.5.4 Aggregate Functions in SQL

Following aggregate functions are provided by the SQL.

1. COUNT - Returns number of tuples.
2. SUM - Returns sum of entries in a column.
3. MAX - Returns Maximum value from an entire column.
4. MIN - Returns Minimum value from an entire column.
5. AVG - Returns Average of all the entries in a column.

### 1.5.5 Constraints in SQL

Following constraints are provided by the SQL.

1. NOT NULL - Column should contain some value.
2. PRIMARY KEY - Should not allow duplicate and null values to a column.
3. UNIQUE - Each value of a column should be unique.
4. DEFAULT - Provides a default value for a column when none is specified.
5. [FOREIGN Key](https://www.tutorialspoint.com/sql/sql-foreign-key.htm) - Uniquely identifies a row/record in any of the given database table.
6. CHECK - The CHECK constraint ensures that all the values in a column satisfies certain conditions.
7. [INDEX](https://www.tutorialspoint.com/sql/sql-index.htm) - Used to create and retrieve data from the database very quickly.

### 1.5.6 Triggers in SQL

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

**LITERATURE REVIEW**

**2.1 Analysis of the Existing System**

The current system uses a manual grocery management system, from stocks, products, ordering and purchases etc. recorded in a book. This is faced with errors, incompleteness, and insufficient data for analysis. Information regarding stocks, products, sales and purchases are still in black and white which is not properly organized and managed. From the wholesalers to retailer bills, tickets, vouchers, receipts of products are recorded in a book but further operations are not being properly handled. As a result it is difficult in processing, updating and managing.

**2.1.1 The Factors for these difficulties are**

* **Labor-Intensive**

A manual Grocery Management System can be highly labor-intensive to operate. They require continuous monitoring to ensure that each transaction is accounted for and that products are maintained at the appropriate stocking levels. It is also more difficult to share inventory information throughout the business, because the lack of computerization makes accessing inventory records a more cumbersome process. The time spent monitoring inventory levels could be used on more productive activities for the business.

* **Human Error**

A manual Grocery management system relies heavily on the actions of people, which increases the possibility of human error. People might forget to record a transaction or simply miscount the number of goods. This results in needless additional orders that increase the company’s inventory carrying costs and use up precious storage space. Inaccurate physical counts could also result in not ordering enough of a product, meaning the business could run out of a crucial item at the wrong time.

* **Time Wasting**

A manual Grocery management system has a huge tendency of time wasting as the sales manager could have a lot to tackle while many customer seeks attention and this is really affecting the business.

**2.2 Analysis of the New System**

To reduce the shortcomings of the existing system there is a need to develop a new system that could upgrade the status of the current system which is manual and slow to the system that will be automatic and fast. The new system should be concern with offering the requirements of the customer and the workers, the system should be reliable, easier, fast, and more informative. The new system should possess the qualities stated below.

**2.2.1 Qualities of the new System**

* Reduction in processing cost.
* Error reduction
* Automatic posting.
* Improve reporting.
* Automatic production of bills.
* Faster response time.
* Ability to meet user requirements.
* Flexibility.
* Reduced dependency.
* Improve resource use.
* Reduction on use of manpower.
* Reduction in use of paper

# 2.3 Software Used

## 2.3.1 Software Requirements

* + - * **OS:** Windows

### 2.3.2 Software used

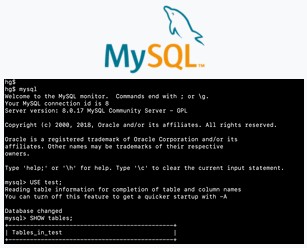
### Backend: Python.

* + - * **Database:** MySQL(sqlite).
      * **Front End:** Python(TKinter).

## 2.3.3 Overview of Tools/Software

* **MySQL**

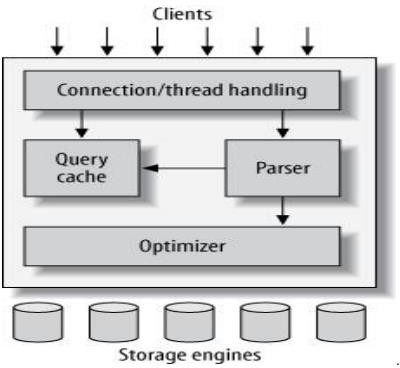
MYSQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co- founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmer use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups



**Screenshot of the default MySQL command-line banner and prompt**

### MySQL’s Logical Architecture:

The topmost layer contains the services that aren’t unique to MySQL. They’re services most network-based client/server tools or servers need: connection handling, authentication, security, and so forth.



**MySQL’s Logical Architecture**

The third layer contains the storage engines. They are responsible for storing and retrieving all data stored “in” MySQL. Like the various file systems available for GNU/Linux, each storage engine has its own benefits and drawbacks. The server communicates with them through the storage engine API. This interface hides differences between storage engines and makes them largely transparent at the query layer. The API contains a couple of dozen low- level functions that perform operations such as begin a transaction” or “fetch the row that has this primary key. The storage engines don’t parse SQL or communicate with each other; they simply respond to requests from the server.

## TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.



**Tkinter Widgets**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [Button](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [Entry](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [Frame](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 6 | [Label](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| 8 | [Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| 9 | [Menu](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [Message](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | [Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [Scale](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| 13 | [Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [Text](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 15 | [Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container. |
| 16 | [Spinbox](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | [PanedWindow](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | [LabelFrame](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | [tkMessageBox](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

## Standard attributes

## 

* Dimensions
* Colors
* Fonts
* Anchors
* Relief styles
* Bitmaps
* Cursors

**Geometry Management**

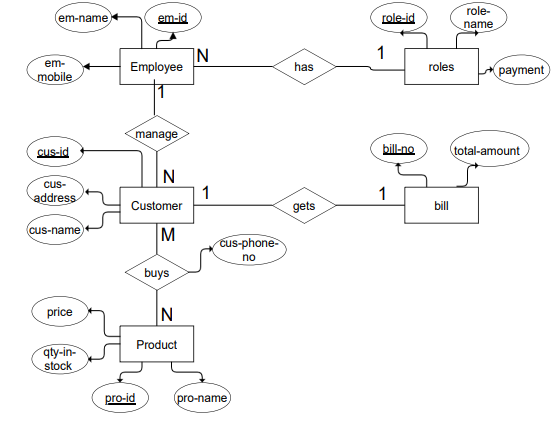
All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
* The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
* The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

# PROPOSED WORK

# 

# 3.1 ER Diagram for Grocery Shop Management System



**ER Diagram for Grocery Management System**

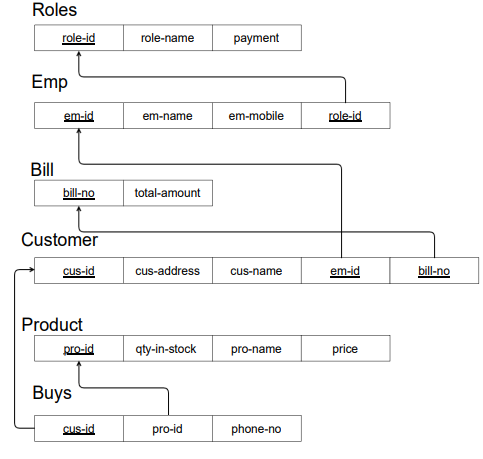
### ER Diagram for Grocery Shop Management system Explanation

The system that we have implemented has 5 entities. They are as follows:

1. **Employee:** This entity contains information about the employee. The primary attribute is the Employee id which is underlined.
2. **Roles:** This entity contains the list of all the roles an employee can have in the grocery shop. Role id is the primary key in this entity.
3. **Customer:** Designed to contain the basic details of the person who buys at the grocery shop. These include name, phone number, address. Customer id is the primary key
4. **Bill:** When a customer makes a purchase, a bill is generated. The bill number and total amount spent by the customer is kept in this entity. Bill number is the primary key.
5. **Product:** This entity contains the name, price and quantity in stock of all the products sold in the grocery shop. Product id is the primary key in this entity.

* The relationship “manage” has a cardinality ratio of 1: N, since an employee can manage ‘n’ customers.
* The relationship “gets” has a cardinality ratio of 1:1, since a customer gets only 1 bill containing all items purchased.
* The relationship “buys” has a cardinality ratio of M: N, as one customer can buy multiple products.
* The relationship “has” has a cardinality ratio of N:1, as n employees can have same role.

## RELATIONAL DATABASE SCHEMA



**Relational Database Schema for Grocery Management System**

### Relational Mapping

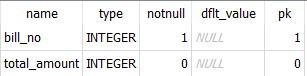
A relational schema for a database is an outline of how data is organized. It can be a graphic illustration or another kind of chart used by programmers to understand how each table is laid out, including the columns and the types of data they hold and how tables connect. A database schema usually specifies which columns are primary keys in tables and which other columns have special constraints such as being required to have unique values in each record. In this section we describe the steps of an algorithm for ER to relational mapping. The above ER diagram is used to derive the respective relational schema. The mapping will create tables with simple single valued attributes.

The steps are:

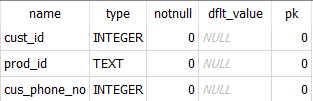
* + - * Mapping of Regular Entity Types.
      * Mapping of Weak Entity Types.
      * Mapping of Binary 1: 1 Relationship Types.
      * Mapping of Binary 1: N Relationship Types.
      * Mapping of Binary M : N Relationship Types.
      * Mapping of Multivalued Attributes.
      * Mapping of N-ary Relationship Types.
* **Mapping of 1: N Relationship type:** For each regular binary 1: n relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as the foreign key in S the primary key of relation T that represents other entity type in R. We have to do this because each entity instance on n side is related to at most one entity instance on 1-side of relationship type.
* **Mapping of M: N Relationship type:** For each m:n relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity type, their combination will form the primary key of S. Also include any simple attributes of m: n relationship type.
* **Mapping of 1:1 Relationship type:** For each 1:1 relationship type R can be migrated to any participating entity types. This relationship type ensures that each user in the database can lodge one complaint

## Database Tables/Relations

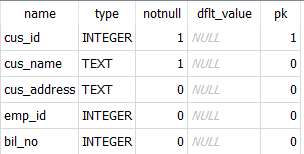
### BILL TABLE



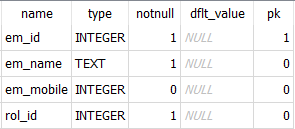
**BUYS TABLE**



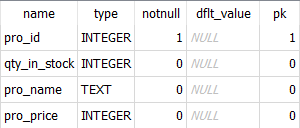
### CUSTOMER TABLE



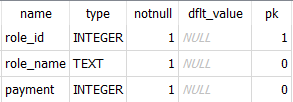
**EMPLOYEE TABLE**



### PRODUCT TABLE

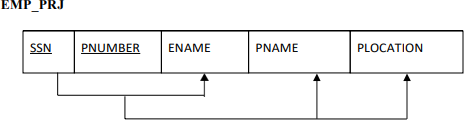


**ROLES TABLE**



## NORMALIZATION OF RELATIONS

### Functional Dependency

The Functional Dependency denoted by X →Y, between two sets of attributes X and Y that are subsets of R species a constraint on the possible tuples that can form a relation stare r of R. The constraints is that for any two tuples t1 and t2 in r that have t1[X] = t2[X]. This means that the values of the Y component of a tuple in r depend on, or determined by the values of the X components. Alternatively, the values of the X component of a tuple uniquely determine the values of the Y component. Consider the following schema,

**Example of Functional Dependency**

In the above schema the functional dependencies are:

1. SSN → ENAME The value of an employee’s social security number (SSN) uniquely determines the employee name (ENAME).
2. PNUMBER → {PNAME, PLOCATION} These values of project’s number uniquely determine the project name (PNAME) and project locations (PLOCATIONS).

### Normalization

The normalization process was proposed by Codd, it takes a relation schema through a series of tests to certify whether if satisfies a certain normal form. The process proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary can thus be considered a relational design by analysis. Normalization of data can be looked upon as a process of analysing the given relation schemas based on their Functional Dependencies and primary keys to achieve the desirable properties of:

* Minimizing redundancy.
* Minimizing the insertion, deletion, and update anomalies.

Normal form of a relation refers to the highest normal form condition that meets, and hence indicates the degree to which it has been normalized.

### Definitions of keys and Attributes Participating in Keys

* **Super Key:** A super key of a relation schema R= {A1, A2… An} is asset of attributes S C R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]. A key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key anymore.
* **Candidate Key:** If a relation schema has more than one key, each key is called candidate key.
* **Prime Attribute:** An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R.
* **Non-Prime Attribute:** An attribute is called nonprime if it is not a prime attribute or it is not a member of any candidate.

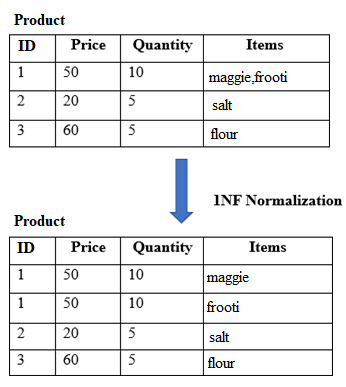
### Normal Forms

There are three normal forms

* First normal form
* Second normal form
* Third normal form

These were proposed by Codd as a sequence to achieve the desirable state of 3NF relations by progressing through the intermediate states of 1Nf and 2NF if needed.

* **First Normal Form (1NF):** It states that the domain of attribute must include only atomic be values and that the value of any attribute in a tuple must be a single value from the domain of the attribute. Hence 1NF disallows having a set of values a tuple of values or a combination of both as an attribute value for a single tuple. In other words, 1NF disallows relations within relations or relations as attribute values within tuples.

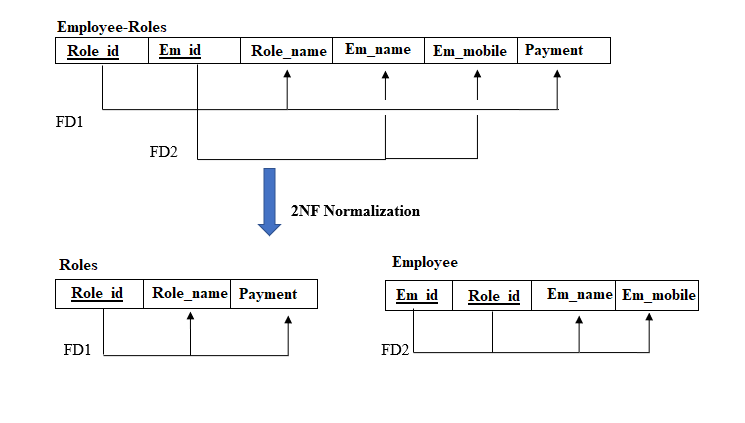


**1NF version of the Products relation**

* **Second Normal Form (2NF):** This normal form is based on the full functional dependency. A functional dependency X →Y is full functional dependency if removal for any attribute A from X means that dependency does not hold any more.

A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R

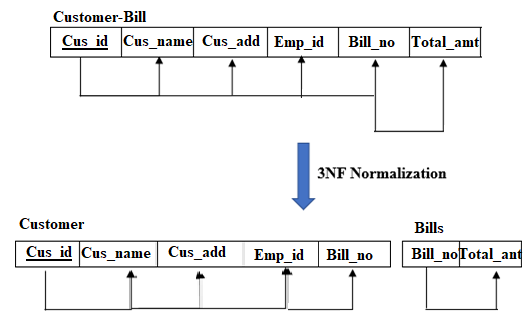
If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent. The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.



**Normalizing Employee\_Roles into 2NF**

* Third Normal Form: Third normal form (3NF) is based on the concept of transitive dependency. A functional dependency X→Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X→Z and Z→Y hold.

A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.

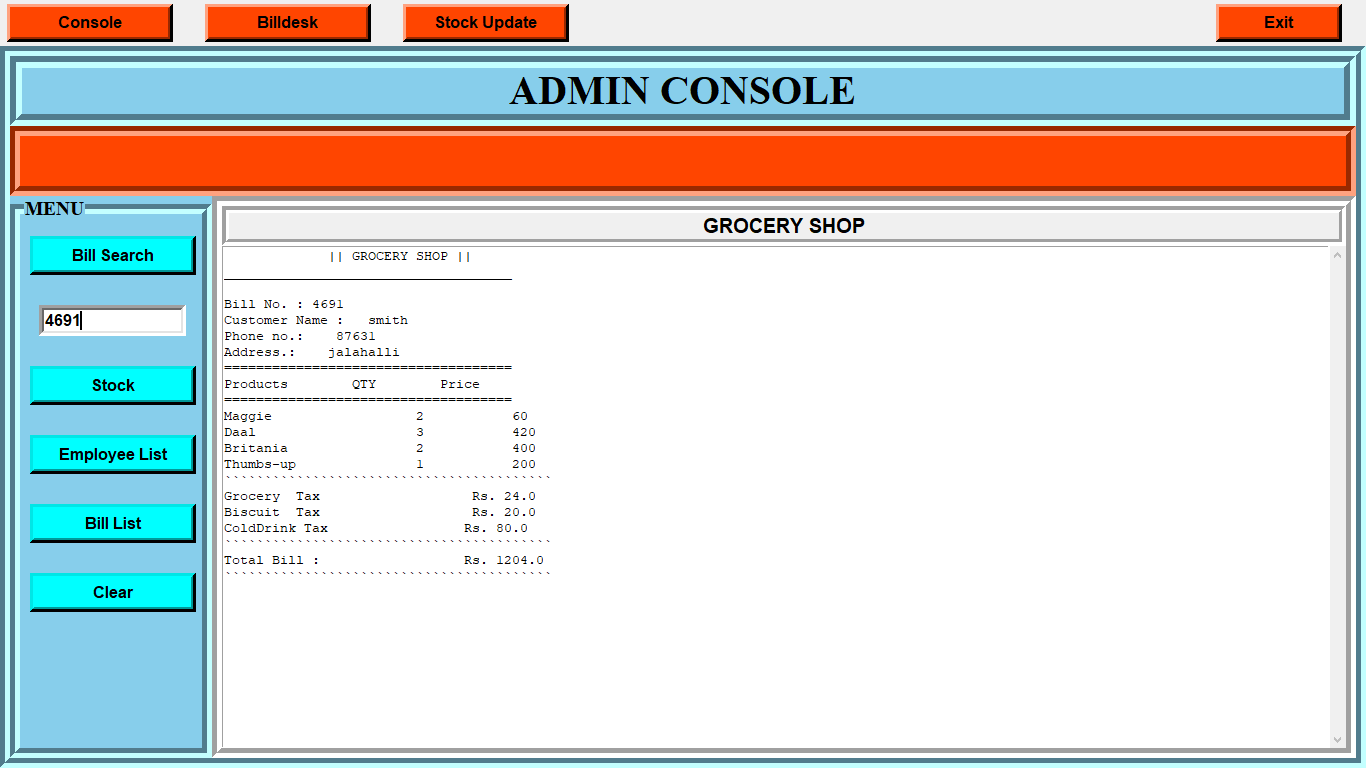


**Normalizing Customer\_Bill into 3NF**

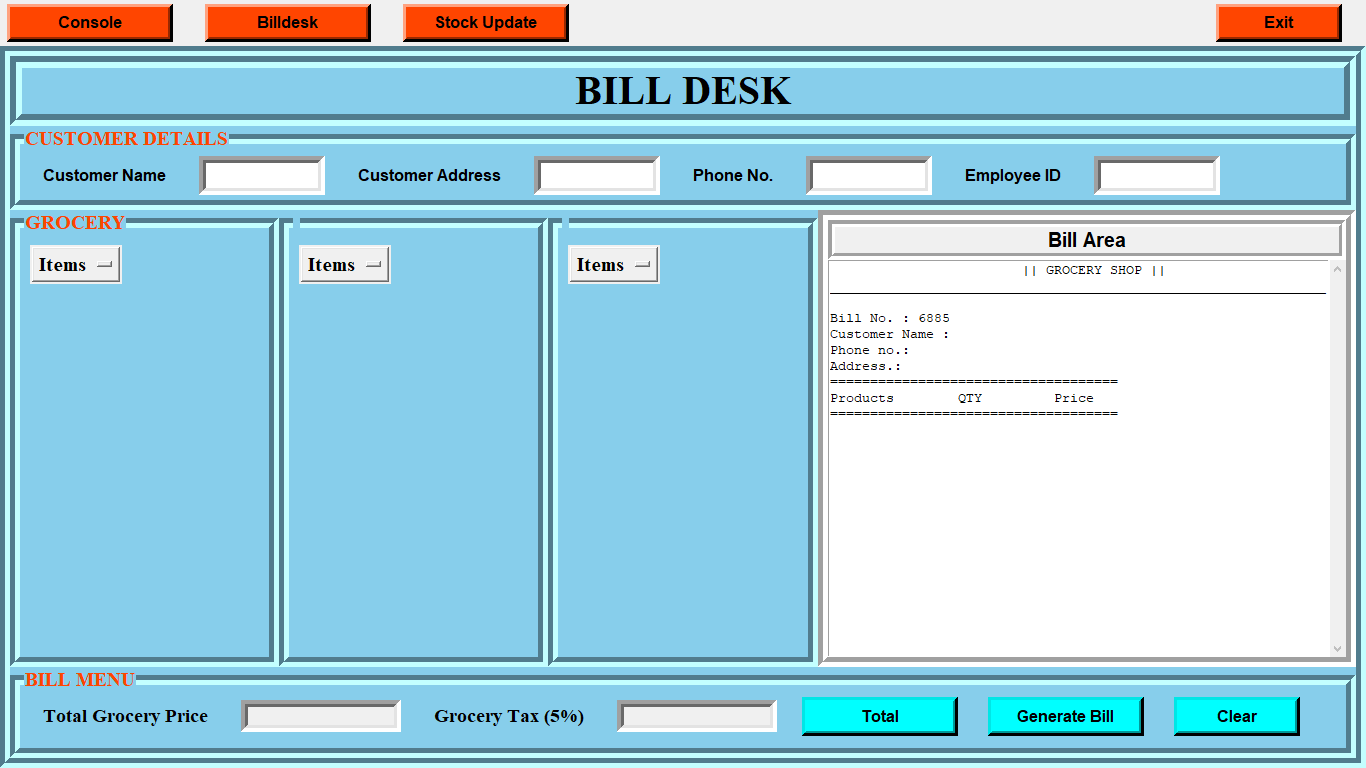
# RESULTS



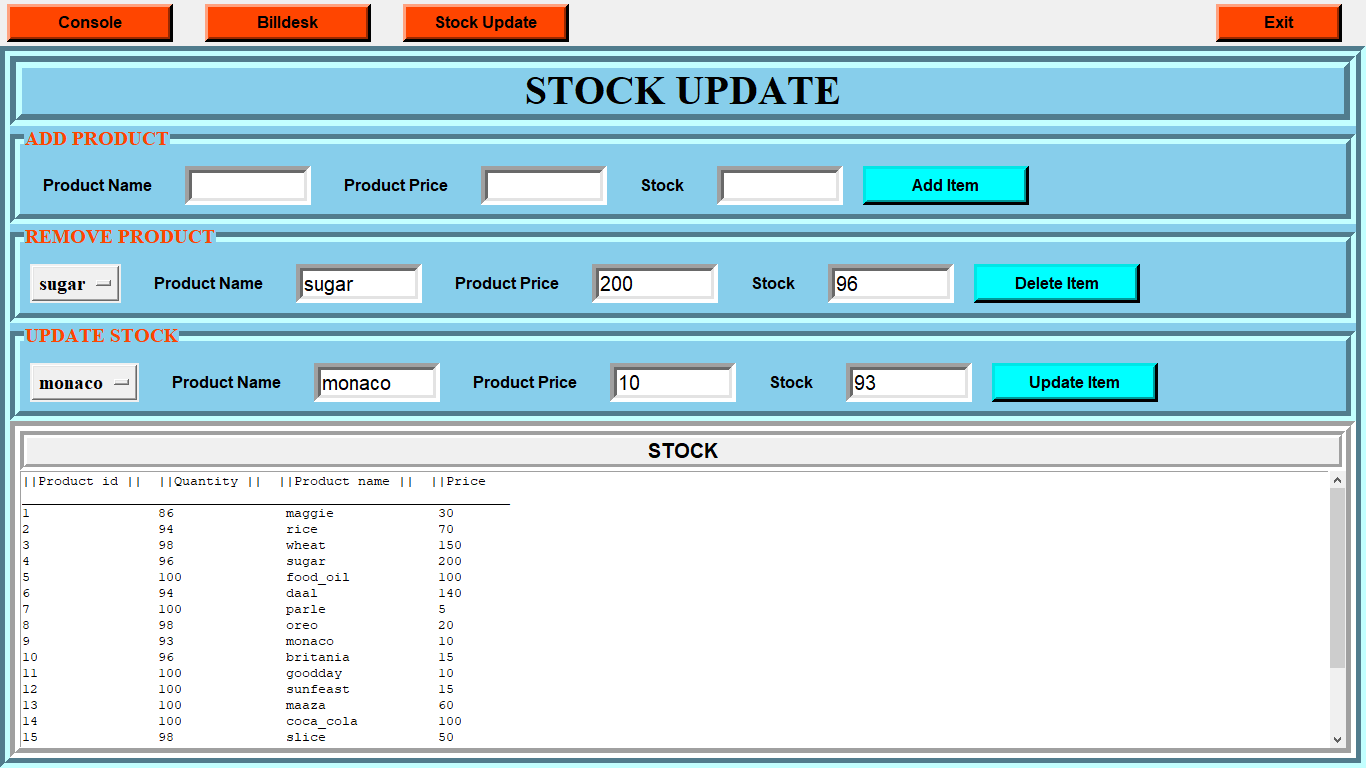
**Homepage**



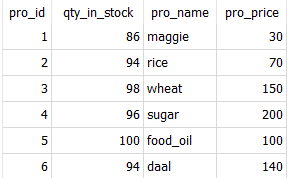
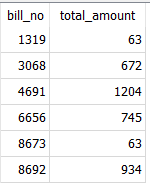
**Admin Console Page**



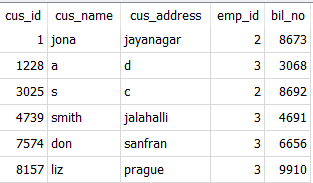
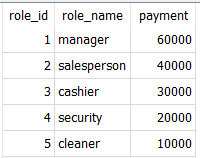
**Bill Desk Page**



**Modify, add, deleting items from product table**

**Product and Bills Tables**



**Roles and Customer Tables**

# CONCLUSION AND FUTURE WORK

## CONCLUSION

This project is carried out to analyse the application of database management through the use of MySQL and the features of Tkinter and the way it can be used and interfaced with MySQL as a standalone database application.

This project was made to make it easier for the employees to maintain stock and generate bills. So, this application is designed to promote Grocery Shops industry.

We were successful in separating stock updation and bill generation side of the application. We made sure that the grocery shop stores the bill details of all the products sold. Easy updation and deletion of products makes the project useful.

We’ve made sure that the website looks simple to make it easy to understand and quick to operate by employees.

## FUTURE WORK

Our project has wide scope of further improvements such as addition of online payment gateway, add delivery option, large scale deployment, containerization. We hope to create master and slave database structure to reduce the overload of the database queries. We also hope to implement combo offers to encourage the customers to buy more products.

**REFERENCES**

* Ramez Elmasri and Shamkant B Navathe, Fundamentals of Database Systems, Fifth Edition, AddisonWesley, 2012.
* Database System Concepts, Sixth Edition, Abraham Silberschatz, Henry F. Korth,

S. Sudarshan : Tata McGraw-Hill, 2010.

* An Introduction to Database Systems by C.J. Date, A. Kannan, S. Swamynathan, 8th Edition, Pearson Education, 2006.
* Database Systems: The Complete Book, Second Edition, Hector Garcia- Molina,Jeffrey D. Ullman, Jennifer Widom, Pearson Education, 2001.
* https://www.w3schools.com/python/
* https://www.w3schools.com/python/python\_mysql\_getstarted.asp
* https://www.tutorialspoint.com/mysql/
* https://docs.python.org/3/library/tkinter.html
* [https://cs.uwaterloo.ca/~tozsu/courses/CS338/lectures/4%20Basic%20SQL.pdf](https://cs.uwaterloo.ca/~tozsu/courses/CS338/lectures/4 Basic SQL.pdf)
* www.cis.gsu.edu/dmcdonald/cis3730/SQL.pdf
* <https://www.geeksforgeeks.org/python-gui-tkinter/>