

SUPERMARKET MANAGEMENT SYSTEM

REVIEW REPORT

Submitted by

PRASON POUDEL (19BCE2550)

MICKEY KUMAR ROUNIYAR (19BCE2520)

SASHANK RIJAL (19BCE2484)

Prepared For

**DATABASE MANAGEMENT SYSTEM (CSE2004)
PROJECT COMPONENT**

Submitted To

Dr. Priya M

Associate Professor

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents:

Chapter

Abstract

1. Introduction

1.1 Background

1.2 Objective

1.3 Motivation

1.4 Contributions of the Project

1.5 Organization of the Project

2. Project Resource Requirements

2.1 Software Requirements

2.2 Hardware Requirements

3. Literature Survey

4. Design of the Project

4.1 ER Diagram

4.3 ER to Relational Mapping (Schema Diagram)

4.4 Normalization

4.5 Tables and Constraints

5. Implementation

5.1 Introduction

5.2 DDL & DML Queries

5.3 SQL Queries

5.4 PL/SQL

6. Screenshot (front end-with explanation)

7. Conclusion and Future Work

7.1 Conclusion

7.2 Future Work

Abstract:

It is very essential to store, organize and make quickly available the necessary data required for the well organized and effective functioning of any business. From a small proprietary business like a simple shop to multi-billion dollar companies like Google, Facebook, etc, , it is really necessary to organize, tabulate, store and secure the data in an efficient manner.

A database management system can help address employee count scenario and a range of other complex situations related to cost, demand and inventory/stock management by presenting the same data to everyone in the business at the same time. Using the basic ideas about database and database management languages,

We want to design a Supermarket management System that will help the shopkeeper to keep track of his stocks and will also provide insights about the business to the shopkeeper based on the stocks, demands, sales, profit or loss.

1. Introduction

1.1 Background:

Using the basic ideas about database and database management languages we learn in this course of Database Management Systems, we want to design a stock keeper for a shop that will help the shopkeeper to keep track of his stocks and will also provide insights about the business to the shopkeeper based on the stocks, demands, sales, profit or loss.

1.2 Objective:

In general, our objective is to include the following features on our application:

- keep a record of stock in the shop
- calculate profit or loss
- notify the shop owner when any item has crossed its expiry date
- Aware the shop owner if an item is trending
- keep record of the employee in the shop
- provide yearly review such as which item was sold more in which month.

1.3 Motivation:

It is very essential to store, organize and make the stored data quickly available for effective functioning of any business. From a small proprietary business like a simple shop to multi-billion dollar companies like Google, Facebook, etc., , it is really necessary to organize, tabulate, store and secure the data in an efficient manner.

If we look into our community it is a general perspective that the software for keeping track of stocks (inventory) should only be used by high end companies and so on. We can also see many stock keeping software are used in big shops such as supermarkets. But, we often see that the track of stocks is often kept manually or sometimes not kept in the smaller shops. So, looking at the scenario we have decided to create a stock keeping software that can run on a very low specifications and would be suitable for a smaller shop. It will help the shops to get insights on their inventory and help for their betterment.

1.4 Contributions of the project:

Register number	Name	Contribution
19BCE2550	Prason Poudel	Front end design and partial Back end code
19BCE2520	Mickey Kumar Rouniyar	Partial Back end code and Database connectivity
19BCE2484	Sashank Rijal	Documentation, Database design and partial front end design

1.5 Organization of the project:

Layout of the application:

Here we will create the user interface. We are going to create it in python with the help of PYQT. PYQT is an external module in python and is nowadays used more than tinkter. The user interface will be made as interactive as possible. It will be a combination of text fields, buttons and small display boxes.

Stock Tracker:

As soon as the shopkeeper buys some things then it will be updated in the database. The items bought has to be manually entered into the system. Similarly, as soon as any item is sold it will be reduced from the database.

Trending/Not to invest item alerter:

In this project, we describe trending item as the item that is being rapidly sold and we define not to invest item as the item that is in the stock for a very long time and is barely sold. This application will be smart enough to classify the products into trending or not. According to the classification, the application will suggest the shopkeeper if the shopkeeper should increase the stock of items or not.

Profit/loss generator

Every owner should be made aware about his /her business. He /She should know whether he is in profit or in loss so that they can take necessary steps to make their business more fruitful. So, one of the modules in our application will calculate daily profit and loss which can be checked by owner. Similarly, it will calculate the yearly profit or loss.

Invoice Creator

It will be better if the stock management system has an invoice generator. If this happens, the Shopkeeper doesn't have to manually enter the record of sales daily. So, Invoice creator will create the bills and with the help of stock tracker module, it will modify the database. Similarly, All the generated bills will be stored for future reference.

Expiry date checker

It is a tedious work to check the expiry date of the products in a shop. It requires extra man-power and money. As our application is already keeping record of all the products, it will also keep track of the expiry date and will check with the system calendar and automatically notify the users.

Employee

It is obvious that a shop will have employees. So, a system is needed to keep the data of the employee and to check if his/her salary is paid or not. This feature will be enbuilt in our sys-tem.

Login

There will be owner and employees in the shop. Each will have their own login credentials. 11.9 Yearly review provider It will access data from all the above modules and will mention which month was the most profitable, which item to be sold on which month.

2. Project Resource Requirement:

2.1 Software Requirements

Operating System:

1. Windows 7 with server pack 2 (or)
2. Windows 8.1 or higher (or)
3. Linux / MacOS

Software's/Modules:

1. Python IDE
2. PyQt python module
3. DB Browser for SQLite2

2.2 Hardware Requirements

Processor: Intel Core Duo 2.0 GHz or more

RAM: 1 GB or More

Hard disk: 50GB or more

Monitor: 15" CRT, or LCD monitor

Keyboard: Normal or Multimedia

Mouse: Compatible mouse

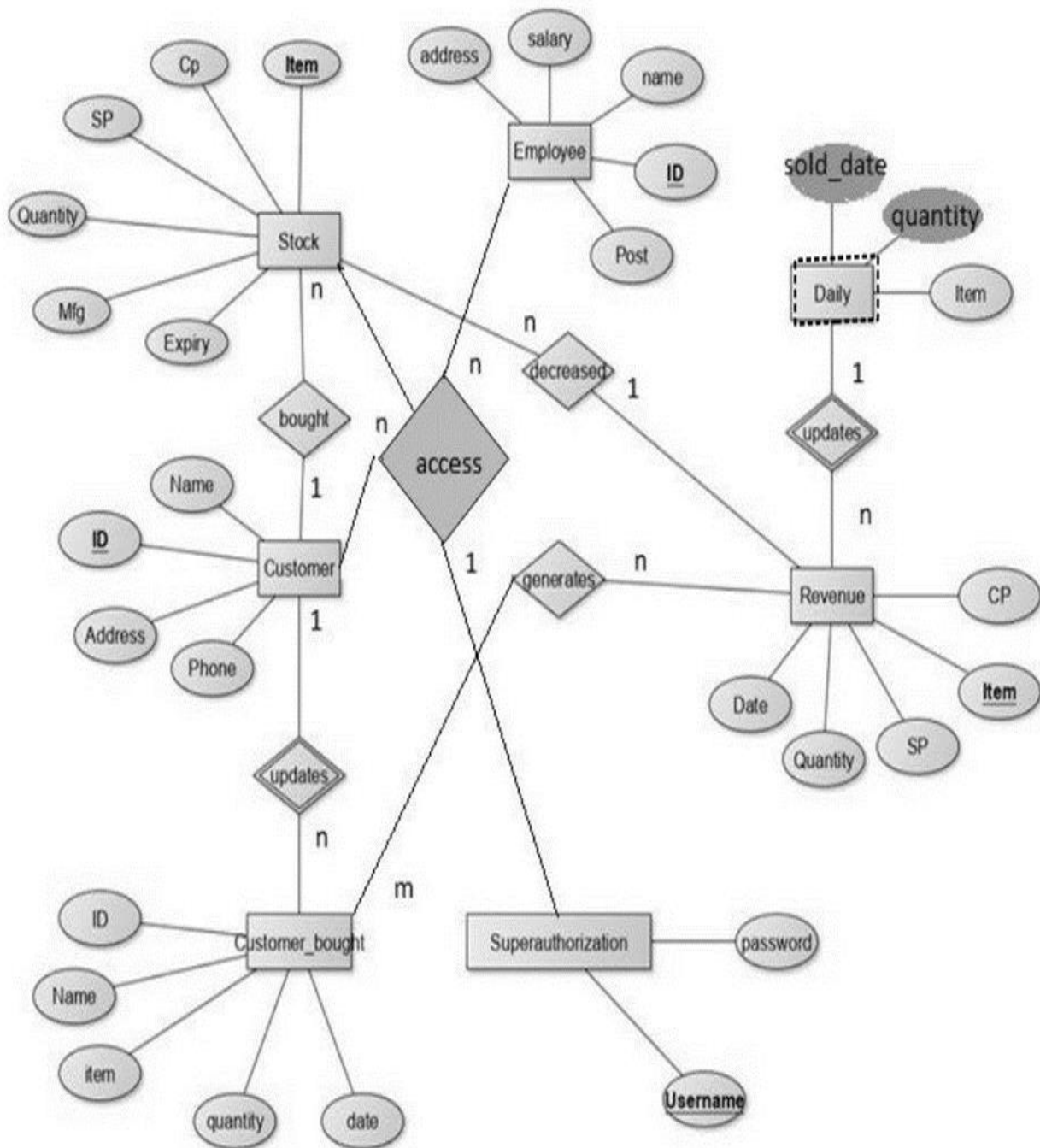
3. Literature Survey:

Authors	Method	Purpose	Advantages	Disadvantages
R. Elmasri & S. B. Navathe	Implementing SQL in Python	Information on Relational Database and levels of Database	Helps to understand implementing sql in python	Requires external pyqt modules.
Thomas Connolly, Carolyn Begg	Database Design	Basic Concept on how to design an efficient database.	Helps in efficient database design.	Lacks about the information on programming the database.
Mark Summerfield	Programming and implementation.	Reference for a programmer	Helps to learn Graphical interface programming	Very lengthy code is generated .
Moskowitz, Robert (Retrieved August 17, 2010)	Client Server Interface.	Using your computer for Inventory Control	Explains the need of digital inventory system	Existence of security issues and vulnerable to hacking.
Lockard, Robert (29 November 2010)	How a business can implement Inventory Management and how it is benefitted.	Use of Inventory Management Software in various applications.	Different businesses are benefitted using the system.	The implementation becomes highly expensive.

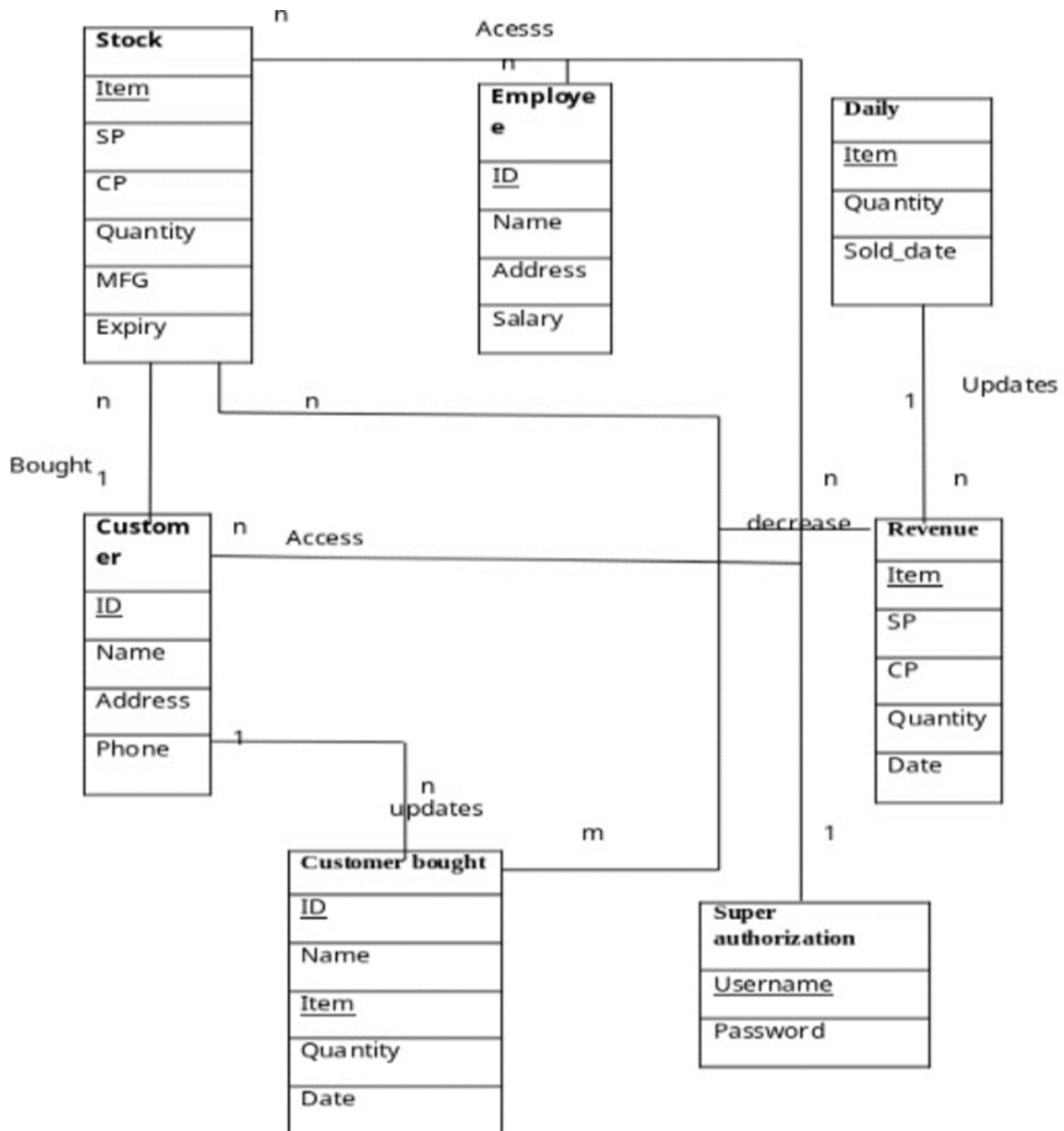
Utkarsha Mendhe, Ankita Lohave, Aayush Sah, Varun Pahwal, Prof. Nutan Sonwane	inventory system was made by them and studied	Bill Generator and Inventory Maintenance	Shows the way of maintaining the inventory in a systematic order in accordance to the software	It adopts a particular way of bill generation that is not suitable for very small data sets
A Rohini, Wubshetasamnew Woldeyohannshiluf and Teshometeklemichea I Walelignayimelo	Studied client server database inventory system	Inventory Management and organisation.	Gives way of organizing the inventory	It sticks to a particular organization system
Lesonsky, Rieva (1998) Entrepreneur Magazine	This paper gives a brief outline on how an inventory should work.	Tracking Inventory	Gives guidelines on how an inventory should work	-
EGA Futura	Case study	Inventory Management Software	It describes how an inventory management software should be	It doesn't talk about the cost of implementing the system

4. Design of the Project

4.1 ER Diagram



4.2 ER to Relational Mapping (Schema Diagram)



4.3 Normalization

Stock

Item	CP	SP	Quantity	Mfg	Expiry
lollipop	12	15	100	03/05/2019	14/08/2020
Noodles	18	25	500	02/09/2019	04/10/2020
Cookies	88	100	400	15/04/2019	16/12/2020
Cricket Bat	75	100	50	22/09/2018	19/12/2023
Pencil	6	8	1000	30/12/2019	30/12/2022
Bed	50000	65000	25	22/12/2019	22/12/2023
Tea leaves	16	20	800	25/09/2019	22/12/2020
Ice Cream	32	50	780	09/04/2019	10/11/2020
Note Book	12	20	500	07/04/2019	07/04/2022
Kitchen Set	440	550	100	05/10/2019	05/12/2024

Functional Dependency

Item \rightarrow CP, SP, Quantity, Mfg, Expiry

This Functional Dependency says that each row has a unique Item. Since this is unique, when provided this value, we can get the all remaining values of that row. In other words, the RHS of this FD is dependent on LHS attribute.

First Normal Form:

First Normal form is a normalization of a table where there are no divisible records and there are no multi-valued attributes. If both the conditions are satisfied, then we can say that the table is in 1NF.

The table above is already in First normal form because the attributes contains atomic values which means none of them are divisible.

Second Normal Form

Second Normal form is a normalization of 1NF table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key. First the table must be normalized to 1NF then the procedure of 2NF can be carried out.

This table has only one primary key that is item. And all non-key attributes are fully functional dependent on it. So this table is in 2NF already.

Third Normal Form

Third Normal form is a normalization where all transitivity are removed. For example, in a table A, attribute $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity. Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.

In Stock table, there is no transitivity.

Item \rightarrow CP, SP, Quantity, Mfg, Expiry

These are functional dependencies of this table. As shown, this cannot be implied in $X \rightarrow Y$ and $Y \rightarrow Z$ formula. Because the RHS of this FD are all non-prime attributes or non-key attributes, hence they are not functionally dependent on anything. So this table clearly does not have transitivity. So the table Employee is in 3NF already and hence no decomposition is required.

BCNF:

BCNF is a normalization where LHS of FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF. For example, in FD $X \rightarrow Y$, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.

In the table Stock the functional dependency is:

Item \rightarrow CP, SP, Quantity, Mfg, Expiry

Item is super key because all RHS attributes are fully functionally dependent on Item (LHS). Moreover, no more FDs can be formed out of this table. Since there is a super key already we can say that this table is in BCNF.

The decomposition is not required as it is already in BCNF.

Employee

Id	Address	Name	Salary	Post
E001	Butwal	Prason	90,000	Product Manager
E002	Duhabi	Mickey	2,00,000	Executive Manager
E003	Biratnagar	Sashank	50,000	Cashier
E004	Kathmandu	Anish	30,000	Stock Clerk
E005	Birtamod	Pramshu	25,000	Custodians
E006	Butwal	Pramshu	20,000	Delivery Driver

Functional Dependency:

Id \rightarrow Address, Name, Salary, Post

First Normal Form:

The table is in 1NF already because the attributes contains atomic values which means none of them are divisible.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfy the rule of BCNF.

Customer

Id	Name	Phone	Address
C001	Bipin	9821964261,9837820323	Shankarnagar
C002	Satkar	9837820172,9837283631	Birtamod
C003	Anubhav	9836826182	Chitwan
C004	Adam	9826825812	Lahore
C005	Bipin	9826826210	Kanchanbadi

Functional Dependency:

Id → Name, Phone, Address

First Normal Form:

The table above is clearly not in 1NF because Phone has multiple values per attributes. This clearly violates the rule of 1NF.

To normalize into 1NF we make separate records for every multiple attributes.

Id	Name	Phone	Address
C001	Bipin	9821964261	Shankarnagar
C001	Bipin	9837820323	Shankarnagar
C002	Satkar	9837820172	Birtamod
C002	Satkar	9837283631	Birtamod
C003	Anubhav	9836826182	Chitwan
C004	Adam	9826825812	Lahore
C005	Bipin	9826826210	Kanchanbadi

Now the new table is normalized into 1NF. For attributes Bipin and Satkar , there were two Phone per attributes. Now for each values there are two separate records. All the attributes are atomic values which means none of them are divisible.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfy the rule of BCNF.

Customer_Bought

Id	Name	Items	Date	Quantity
C001	Bipin	Cookies	02/06/2020	5
C002	Satkar	Noodles	09/05/2020	2
C003	Anubhav	Lolipop	17/09/2020	11
C004	Adam	Bed, Kitchen Set	22/09/2020	1, 1
C005	Bipin	Tea leaves	14/08/2020	5

Functional Dependency:

Id → Name, Items, Date, Quantity

First Normal Form:

The table above is clearly not in 1NF because Phone has multiple values per attributes. This clearly violates the rule of 1NF.

To normalize into 1NF we make separate records for every multiple attributes.

Id	Name	Items	Date	Quantity
C001	Bipin	Cookies	02/06/2020	5
C002	Satkar	Noodles	09/05/2020	2
C003	Anubhav	Lolipop	17/09/2020	11
C004	Adam	Bed	22/09/2020	1
C004	Adam	Kitchen Set	22/09/2020	1
C005	Bipin	Tea leaves	14/08/2020	5

Now the new table is normalized into 1NF. For attribute Adam there was two Item. Now for there are two separate records. All the attributes are atomic values which means none of them are divisible.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfy the rule of BCNF.

Revenue

Item	CP	SP	Profit	Quantity	Date
Cookies	88	100	12	5	02/06/2020
Noodles	18	25	7	6	09/05/2020
Lolipop	12	15	3	11	17/09/2020
Bed	50000	65000	15000	1	22/09/2020
Kitchen Set	440	550	110	1	22/09/2020
Tea Leaves	16	20	4	5	14/08/2020

Functional Dependency:

Item \rightarrow CP, SP, Profit, Quantity, Date

First Normal Form:

The table is in 1NF already because the attributes contain atomic values which means none of them are divisible.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfies the rule of BCNF.

Daily

Item	Quantity	Sold Date
Cookies	2	22/09/2020
Noodles	4	22/09/2020
Lolipop	2	22/09/2020
Bed	1	22/09/2020

Functional Dependency:

Item \rightarrow Quantity, Sold Date

First Normal Form:

The table is in 1NF already because the attributes contain atomic values which means none of them are divisible.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfies the rule of BCNF.

Super Authorization:

UserName	Password
Mickey	Mickey123
Prason	Prason123
Sashank	Sashank123

Functional Dependency:

UserName → Password

First Normal Form:

The table is in 1NF already because the attributes contain atomic values which means none of them are divisible and only two attributes are there.

Second Normal Form:

The table is in 2NF because there is no partial dependency.

Third Normal Form

The table is in 3NF because there is no transitive dependency.

BCNF Form

The table is in BCNF already because the LHS of functional dependency is a super key which satisfies the rule of BCNF.

4.4 Tables and Constraints

Stock:

Column name	Data type	constraint
Item	VARCHAR2(20)	Primary key
CP	NUMBER(10)	NOT null
SP	NUMBER(10)	NOT null
Quantity	NUMBER(10)	NOT null
Mfg	DATE	NOT null
Expiry	DATE	NOT null

Employee:

Column name	Data type	constraint
Id	NUMBER(10)	Primary key,Unique
Address	VARCHAR2(30)	NOT null
Name	VARCHAR2(20)	NOT null
Salary	NUMBER(10)	NOT null
Post	VARCHAR2(20)	NOT null

Customer:

Column name	Data type	constraint
Id	NUMBER(10)	Primary KEY,Unique
Name	VARCHAR2(20)	NOT null
phone	NUMBER(10)	NOT null,check
address	VARCHAR2(30)	NOT null

Customer_bought:

Column name	Data type	constraint
Id	NUMBER(10)	Foreign key
Name	VARCHAR2(20)	NOT null
Items	VARCHAR2(20)	NOT null
Date	DATE	NOT null
quantity	NUMBER(10)	NOT null

Revenue:

Column Name	Data Type	constraint
Item	VARCHAR2(10)	Primary key
CP	NUMBER(10)	NOT null
SP	NUMBER(10)	NOT null
Profit	NUMBER(10)	NOT null
Quantity	NUMBER(10)	NOT null
Date	DATE	NOT null

Daily:

column_name	Data type	constraint
Item	VARCHAR2(20)	Foreign key
quantity	NUMBER(10)	Not null
Sold_date	DATE	Not null

Superauthorization:

column_name	Data type	constraint
Username	VARCHAR2(20)	Primary key,Unique
Password	VARCHAR2(20)	Not null,check

5. Implementation

5.1 Introduction

We often see that the track of stocks is often kept manually or sometimes not kept in the smaller shops with not such a skilled employees. So, looking at the scenario we have decided to create a stock keeping software that can run on a very low specifications and would be suitable for a smaller shop and can be run very easily and virtually by anyone. It will help the shops to get insights on their inventory and help for their betterment.

5.2 DDL & DML Queries

```
create table employee (  
id varchar2(7) constraint employee_pk primary key,  
address varchar2(20),  
name varchar2(20),  
salary number,  
post varchar2(20)  
);
```

```
create table stock (  
item varchar2(20) constraint stock_pk primary key,  
cp number,  
sp number,  
quantity number,  
mfg date,  
expiry date  
);
```

```
create table customer (  
id varchar2(7) constraint customer_pk primary key,  
name varchar2(20),  
phone number,  
address varchar2(20)  
);
```

```
create table customer_bought (  
id varchar2(7),  
constraint ch_fk foreign key(id) references customer(id),  
name varchar2(20),  
item varchar2(20),  
date_date,  
quantity number  
);
```

```
create table daily (
item varchar2(20),
constraint da_fk foreign key(item) references stock(item),
quantity number,
solddate date
);
```

[illegible]

```

insert into stock values('lolipop',12,15,100,'03-May-19','14-Aug-20');
insert into stock values('noodles',18,25,500,'02-Sep-19','04-Oct-20');
insert into stock values('cookies',88,100,400,'15-Apr-19','16-Dec-20');
insert into stock values('bed',50000,65000,25,'22-Dec-19','22-Dec-23');
insert into stock values('kitchen set',440,550,100,'05-Oct-19','05-Dec-24');
insert into stock values('tealeaves',16,20,800,'25-Sep-19','22-Dec-20');

insert into employee values('E001','butwal','prason',90000,'product manager');
insert into employee values('E002','duhabi','mickey',200000,'executive manager');
insert into employee values('E003','biratnagar','sashank',50000,'cashier');
insert into employee values('E004','kathmandu','anish',30000,'stock clerk');
insert into employee values('E005','birtamod','pramshu',25000,'custodians');
insert into employee values('E006','butwal','pramshu',20000,'delivery driver');
insert into customer values('C001','bipin',9821964261,'shankarnagar');
insert into customer values('C002','satkar',9837820172,'birtamod');
insert into customer values('C003','anubhav',9836826182,'chitwan');
insert into customer values('C004','adam',9826825812,'lahore');
insert into customer values('C005','bipin',9826826210,'kanchanbadi');
insert into customer_bought values('C001','bipin','cookies','02-Jun-20',5);
insert into customer_bought values('C002','satkar','noodles','09-May-20',2);
insert into customer_bought values('C003','anubhav','lolipop','17-Sep-20',11);
insert into customer_bought values('C004','adam','bed','22-Sep-20',1);
insert into customer_bought values('C005','bipin','tea leaves','14-Aug-20',5);
insert into revenue values('cookies',88,100,12,5,'02-Jun-20');
insert into revenue values('noodles',18,25,7,6,'09-May-20');

```



```
insert into revenue values('lolipop',12,15,3,11,'17-Sep-20');
insert into revenue values('bed',50000,65000,15000,1,'22-Sep-20');
insert into revenue values('kitchen set',440,550,110,1,'22-Sep-20');
insert into revenue values('tea leaves',16,20,4,5,'14-Aug-20');
insert into customer values('C001','bipin',9821964261,'shankarnagar');
insert into customer values('C002','satkar', 9837820172,'birtamod');
insert into customer values('C003','anubhav', 9836826182,'chitwan');
insert into customer values('C004','adam', 9826825812,'lahore');
insert into customer values('C005','bipin', 9826826210,'kanchanbadi');
insert into customer_bought values('C001','bipin','cookies','02-Jun-20',5);
insert into customer_bought values('C002','satkar','noodles','09-May-20',2);
insert into customer_bought values('C003','anubhav','lolipop','17-Sep-20',11);
insert into customer_bought values('C004','adam','bed','22-Sep-20',1);
insert into customer_bought values('C005','bipin','tealeaves','14-Aug-20',5);
insert into revenue values('cookies',88,100,12,5,'02-Jun-20');
insert into revenue values('noodles',18,25,7,6,'09-May-20');
insert into revenue values('lolipop',12,15,3,11,'17-Sep-20');
insert into revenue values('bed',50000,65000,15000,1,'22-Sep-20');
insert into revenue values('kitchen set',440,550,110,1,'22-Sep-20');
insert into revenue values('tealeaves',16,20,4,5,'14-Aug-20');
insert into daily values('lolipop',2,'22-Sep-19');
insert into daily values('noodles',4,'22-Sep-19');
insert into daily values('bed',1,'22-Sep-19');
insert into daily values('cookies',2,'22-Sep-19');
```

5.3 SQL Queries

List the name of the customers who bought cookies after the month of July

```
select customer.name  
from customer join customer_bought  
on customer.id = customer_bought.id  
where customer_bought.date_ > '01-jul-20';
```

SQL Worksheet

 Clear  F

```
1 select customer.name  
2 from customer join customer_bought  
3 on customer.id=customer_bought.id  
4 where customer_bought.date_>'01-jul-20';
```

NAME

anubhav

adam

bipin

[Download CSV](#)

3 rows selected.

List the name and initial quantity of the item that gives the profit of less than 100.

```
Select stock.item,stock.quantity
from stock join revenue
on stock.item = revenue.item
where revenue.profit < 100;
```

SQL Worksheet

 Clear  Find  Act

```
1 select stock.item,stock.quantity
2 from stock join revenue
3 on stock.item=revenue.item
4 where revenue.profit<100;
```

ITEM	QUANTITY
lollipop	100
noodles	500
cookies	400
tealeaves	800

[Download CSV](#)
4 rows selected.

List the details of the customers who either bought cookies or bought kitchen set.

```
select customer.*
from customer join customer_bought
on customer.id=customer_bought.id
where customer_bought.item in(select item from customer_bought where item in('cookies','kitchen
set'));
```

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 select customer.*
2 from customer join customer_bought
3 on customer.id=customer_bought.id
4 where customer_bought.item in(select item from customer_bought where item in('cookies','kitchen set'));
5
```

ID	NAME	PHONE	ADDRESS
C001	bipin	9821964261	shankarnagar

[Download CSV](#)

Give the name and profit of the item bought by the customer whose name starts with b.

Select revenue.item,revenue.profit
from revenue join customer_bought
on revenue.item=customer_bought.item
where customer_bought.name like 'b%';

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 select revenue.item,revenue.profit
2 from revenue join customer_bought
3 on revenue.item=customer_bought.item
4 where customer_bought.name like 'b%';
```

ITEM	PROFIT
cookies	12
tealeaves	4

[Download CSV](#)

2 rows selected.

Display the total number of customers whose name starts with a and who helped in generating the revenue profit of more than 5000.

Select count(id) from customer where name like 'a%' and id=
(select id from customer_bought where item =
(select item from revenue where profit>5000));

SQL Worksheet

ClearFindActions

1Select count(id) from customer where name like 'a%' and id=
2(select id from customer_bought where item =
3(select item from revenue where profit>5000));
4
5

COUNT(ID)

1

Download CSV

Give the details of the items and profits of each that expires on between 01-jan-2020 to 01-dec-2023.

```
Select stock.*,revenue.profit
From stock join revenue
On stock.name=revenue.name
Where stock.expiry > '01-jan-20' and stock.expiry < '01-dec-23';
```

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 Select stock.*,revenue.profit
2 From stock join revenue
3 On stock.item=revenue.item
4 Where stock.expiry>'01-jan-20' and stock.expiry<'01-dec-23';
5
```

ITEM	CP	SP	QUANTITY	MFG	EXPIRY	PROFIT
cookies	88	100	400	15-APR-19	16-DEC-20	12
noodles	18	25	500	02-SEP-19	04-OCT-20	7
lollipop	12	15	100	03-MAY-19	14-AUG-20	3
tealeaves	16	20	800	25-SEP-19	22-DEC-20	4

[Download CSV](#)

4 rows selected.

Give the daily details of the items which was sold on '22-sep-2020' and had a cost price more than 400.

```
Select daily.*
From daily join revenue
On daily.item=revenue.item
Where daily.solddate='22-sep-19'
and revenue.cp>400;
```

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 Select daily.*
2 From daily join revenue
3 On daily.item=revenue.item
4 Where daily.solddate='22-sep-19'
5 and revenue.cp>400;
```

ITEM	QUANTITY	SOLDDATE
bed	1	22-SEP-19

[Download CSV](#)

Give the details of customers that either bought cookies, noodles or tea leaves before the month of October 2020.

```
Select (customer.id),customer.*
From customer join customer_bought
On customer.id=customer_bought.id
Where customer_bought.date<'01-oct-20'
And customer_bought.item in(select item from customer_bought where item
in('cookies','noodles','tea leaves'));
```

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 Select (customer.id),customer.*
2 From customer join customer_bought
3 On customer.id=customer_bought.id
4 Where customer_bought.date<'01-oct-20'
5 And customer_bought.item in(select item from customer_bought where item in('cookies','noodles','tea leaves'));
```

ID	ID	NAME	PHONE	ADDRESS
C001	C001	bipin	9821964261	shankarnagar
C002	C002	satkar	9837820172	birtamod

[Download CSV](#)

2 rows selected.

Give all the details of the item bought by the customer whose name starts with s and have exactly 5 characters in it.

```
Select stock.*
From stock join customer_bought
On stock.item=customer_bought.item
Where customer_bought.name=
(select name from customer where name like 's_____');
```

SQL Worksheet

[Clear](#)[Find](#)[Actions](#) ▾

```
1 Select stock.*
2 From stock join customer_bought
3 On stock.item=customer_bought.item
4 Where customer_bought.name=
5 (select name from customer where name like 's_____');
6
7
```

ITEM	CP	SP	QUANTITY	MFG	EXPIRY
noodles	18	25	500	02-SEP-19	04-OCT-20

[Download CSV](#)

Give all the details of the item bought by a customer who purchased just 1 item from any of the available items and gave the profit of atleast 10000.

Select stock.*
from stock join customer_bought
on stock.item = customer_bought.item
where customer_bought.item=(select item from revenue where profit>10000)
and customer_bought.quantity=1;

SQL Worksheet

Clear

Find

Actions

1

select stock.*

2

from stock join customer_bought

3

on stock.item = customer_bought.item

4

where customer_bought.item=(select item from revenue where profit>10000)

5

and customer_bought.quantity=1;

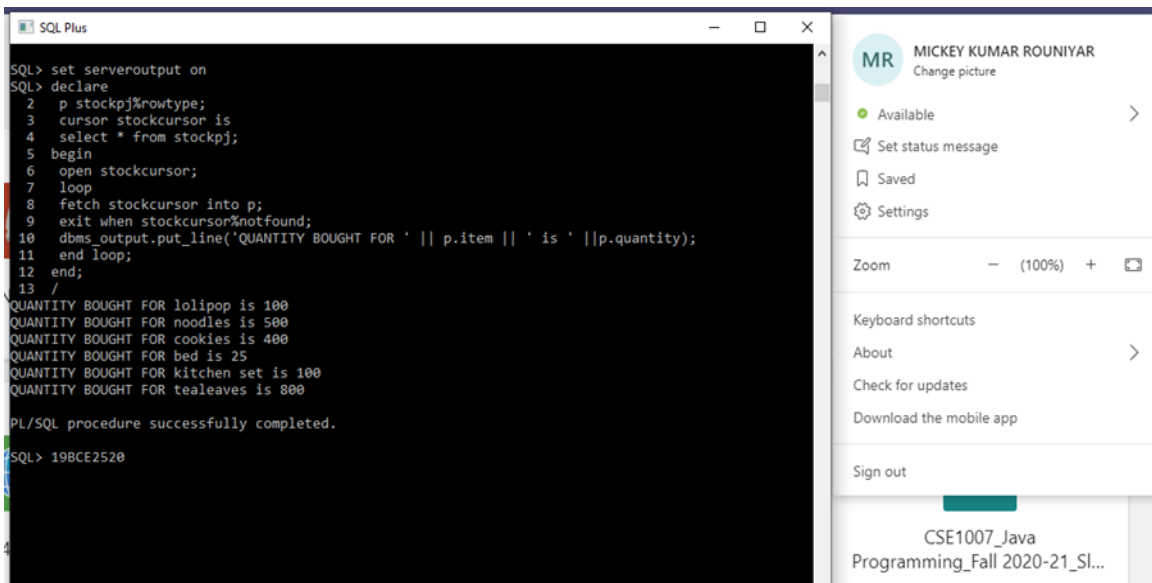
Run SQL Script

ITEM	CP	SP	QUANTITY	MFG	EXPIRY
bed	50000	65000	25	22-DEC-19	22-DEC-23

Download CSV

5.4 PL/SQL

Write a cursor to display all the quantities along with the name of the item from stockpj table.

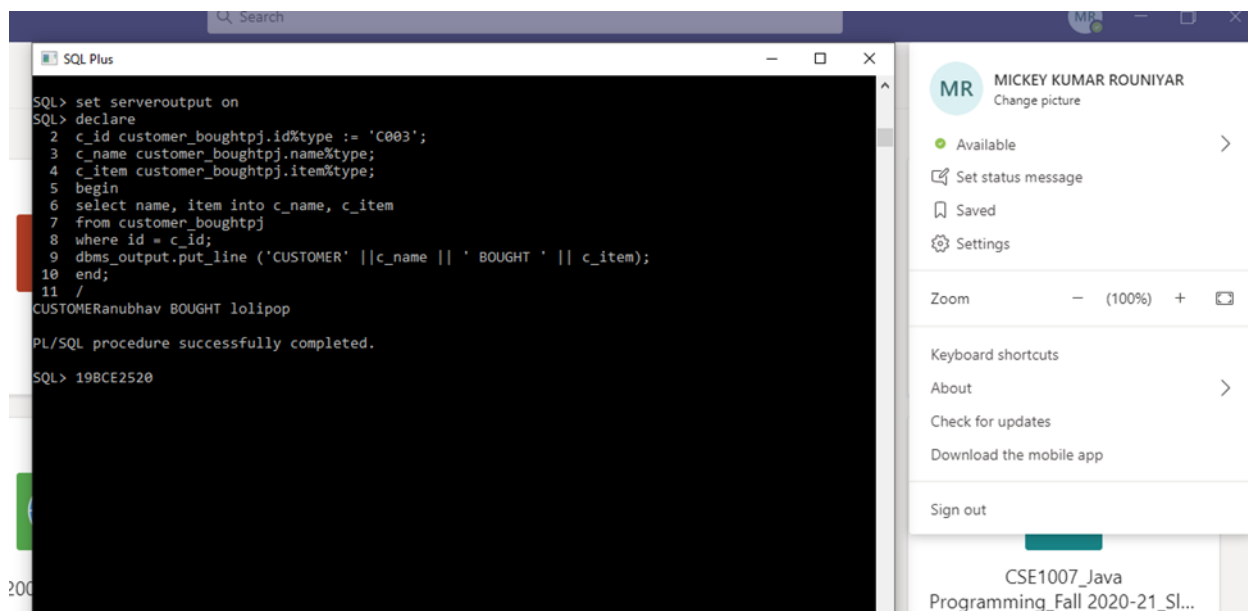


```
SQL> set serveroutput on
SQL> declare
  2  p stockpj%rowtype;
  3  cursor stockcursor is
  4  select * from stockpj;
  5  begin
  6  open stockcursor;
  7  loop
  8  fetch stockcursor into p;
  9  exit when stockcursor%notfound;
 10  dbms_output.put_line('QUANTITY BOUGHT FOR ' || p.item || ' is ' || p.quantity);
 11  end loop;
 12 end;
 13 /
QUANTITY BOUGHT FOR lolipop is 100
QUANTITY BOUGHT FOR noodles is 500
QUANTITY BOUGHT FOR cookies is 400
QUANTITY BOUGHT FOR bed is 25
QUANTITY BOUGHT FOR kitchen set is 100
QUANTITY BOUGHT FOR tealeaves is 800

PL/SQL procedure successfully completed.

SQL> 19BCE2520
```

Write a PL/SQL program to practice reading the record from a table into local variables using different data types and %TYPE and display the same using locally declared variables



```
SQL> set serveroutput on
SQL> declare
  2  c_id customer_boughtpj.id%type := 'C003';
  3  c_name customer_boughtpj.name%type;
  4  c_item customer_boughtpj.item%type;
  5  begin
  6  select name, item into c_name, c_item
  7  from customer_boughtpj
  8  where id = c_id;
  9  dbms_output.put_line ('CUSTOMER' || c_name || ' BOUGHT ' || c_item);
 10 end;
 11 /
CUSTOMERanubhav BOUGHT lolipop

PL/SQL procedure successfully completed.

SQL> 19BCE2520
```

SQL Worksheet

Clear

Find

Actions

Save

Run

```
1 DECLARE
2   NAME employee.post%TYPE;
3   AMOUNT employee.salary%TYPE;
4 BEGIN
5   SELECT post,salary INTO NAME, AMOUNT
6   FROM employee WHERE ID = 'E001';
7   DBMS_OUTPUT.PUT_LINE('SALARY OF ' || NAME || ' IS ' || AMOUNT);
8 END;
9
```

Statement processed.
SALARY OF product manager IS 90000

SQL Worksheet

Clear

Find

Actions

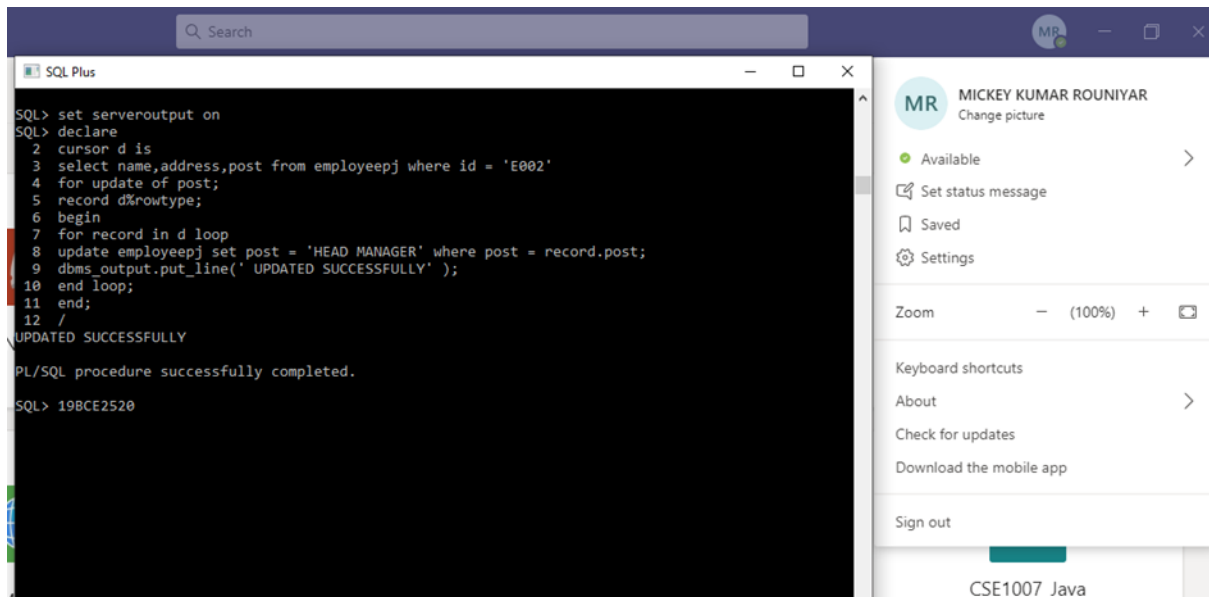
Save

Run

```
1 DECLARE
2   P customer%ROWTYPE;
3   CURSOR PRODUCTCURSOR IS
4   SELECT * FROM customer;
5 BEGIN
6   OPEN PRODUCTCURSOR;
7   LOOP
8     FETCH PRODUCTCURSOR INTO P;
9     EXIT WHEN PRODUCTCURSOR%NOTFOUND;
10    DBMS_OUTPUT.PUT_LINE('CONTACT NUMBER OF ' || P.name || ' who is from ' || P.address || ' is ' || p.phone);
11  END LOOP;
12  CLOSE PRODUCTCURSOR;
13  END;
```

Statement processed.
CONTACT NUMBER OF bipin who is from shankarnagar is 9821964261
CONTACT NUMBER OF satkar who is from birtamod is 9837820172
CONTACT NUMBER OF anubhav who is from chitwan is 9836826182
CONTACT NUMBER OF adam who is from lahore is 9826825812
CONTACT NUMBER OF bipin who is from kanchanbadi is 9826826210

Write a PL/SQL program using cursor to assign a employee to “HEAD MANAGER” whose employee id is E002.



The screenshot shows the SQL Plus command window with the following PL/SQL code and output:

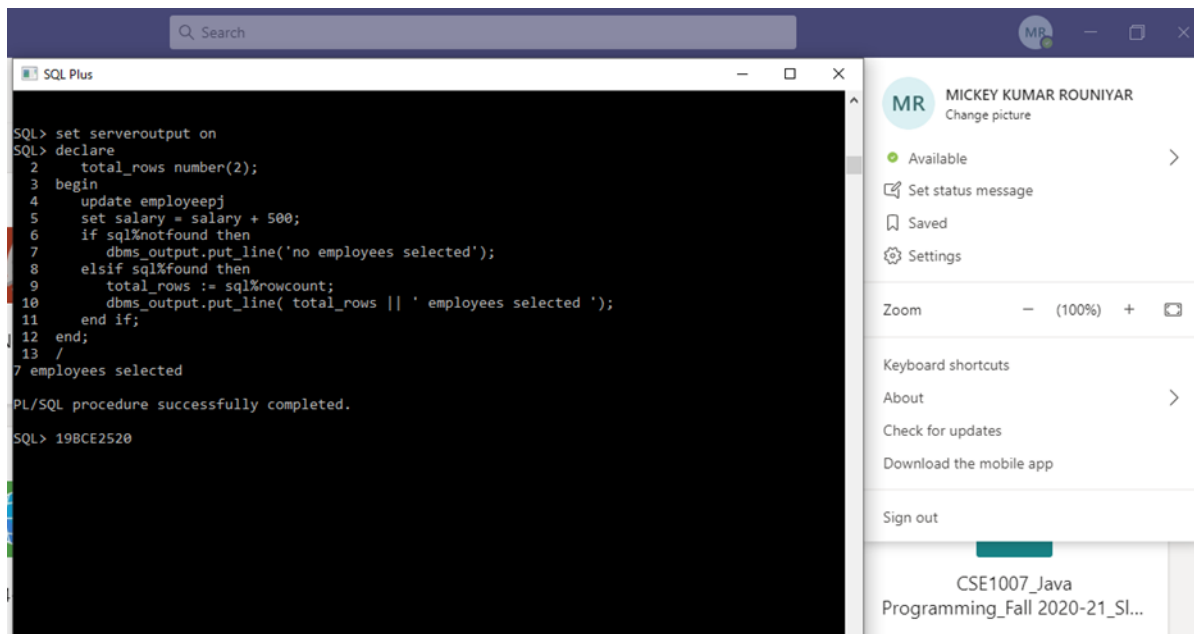
```
SQL> set serveroutput on
SQL> declare
2  cursor d is
3  select name,address,post from employeeej where id = 'E002'
4  for update of post;
5  record d%rowtype;
6  begin
7  for record in d loop
8  update employeeej set post = 'HEAD MANAGER' where post = record.post;
9  dbms_output.put_line(' UPDATED SUCCESSFULLY' );
10 end loop;
11 end;
12 /
UPDATED SUCCESSFULLY

PL/SQL procedure successfully completed.

SQL> 19BCE2520
```

The right sidebar shows the user profile for MICKEY KUMAR ROUNIYAR with options like Available, Set status message, Saved, Settings, Zoom (100%), Keyboard shortcuts, About, Check for updates, Download the mobile app, and Sign out. The bottom of the sidebar shows the file name CSE1007_Java.

Write a PL/SQL program to increase the salary of all the employees and to display the count of the employees whose salary has been increased



The screenshot shows the SQL Plus command window with the following PL/SQL code and output:

```
SQL> set serveroutput on
SQL> declare
2  total_rows number(2);
3  begin
4  update employeeej
5  set salary = salary + 500;
6  if sql%notfound then
7  dbms_output.put_line('no employees selected');
8  elsif sql%found then
9  total_rows := sql%rowcount;
10 dbms_output.put_line( total_rows || ' employees selected ');
11 end if;
12 end;
13 /
7 employees selected

PL/SQL procedure successfully completed.

SQL> 19BCE2520
```

The right sidebar is identical to the first screenshot, showing the user profile for MICKEY KUMAR ROUNIYAR. The bottom of the sidebar shows the file name CSE1007_Java Programming_Fall 2020-21_SI...



SQL Worksheet

Clear

Find

Actions ▾

Save

Run

```
1 CREATE OR REPLACE FUNCTION totalCustomers
2 RETURN number IS
3     total number(2) := 0;
4 BEGIN
5     SELECT count(*) into total
6     FROM customer;
7
8     RETURN total;
9 END;
10 /
11 DECLARE
12     c number(2);
13 BEGIN
14     c := totalCustomers();
15     dbms_output.put_line('Total no. of Customers: ' || c);
16 END;
17 /
```

Function created.

Statement processed.

Total no. of Customers: 6



SQL Worksheet

Clear

Find

Actions ▾

Save

Run

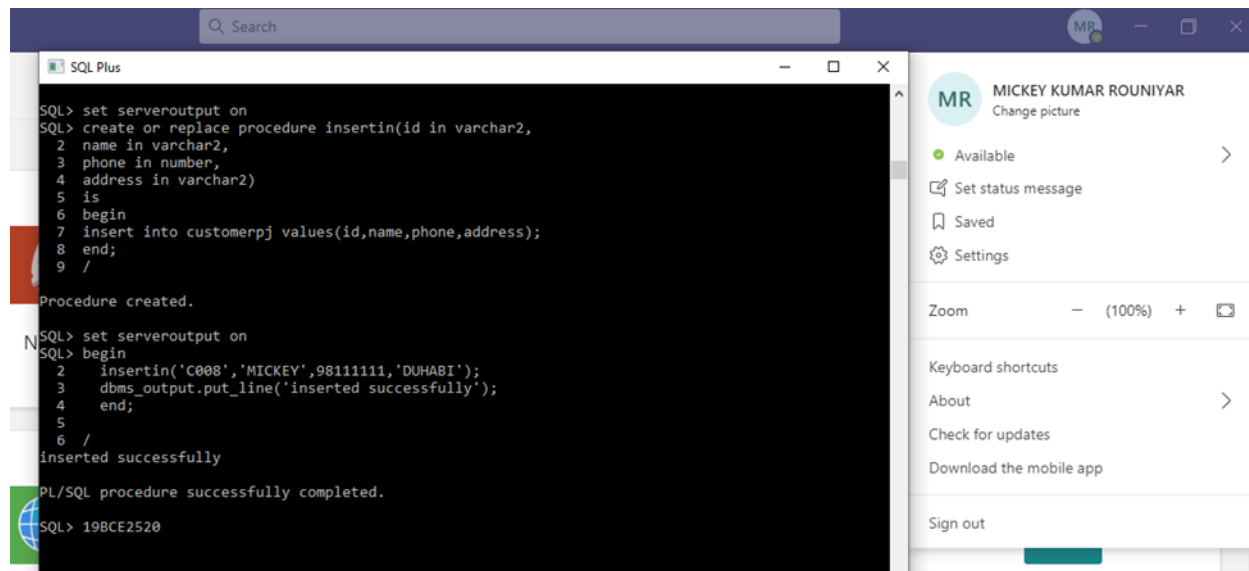
```
1 CREATE OR REPLACE
2 FUNCTION SUB_TWO (cp INT,sp INT)
3 RETURN INT IS
4 BEGIN
5     RETURN (sp-cp);
6 END;
7 /
8 BEGIN
9     DBMS_OUTPUT.PUT_LINE('PROFIT IS: ' || SUB_TWO(12,34));
10 END;
11
```

Function created.

Statement processed.

PROFIT IS: 22

Write a PL/SQL program using procedures to insert values in one of the tables.



The screenshot shows the SQL Plus command window with the following text:

```
SQL> set serveroutput on
SQL> create or replace procedure insertin(id in varchar2,
2 name in varchar2,
3 phone in number,
4 address in varchar2)
5 is
6 begin
7 insert into customerpj values(id,name,phone,address);
8 end;
9 /

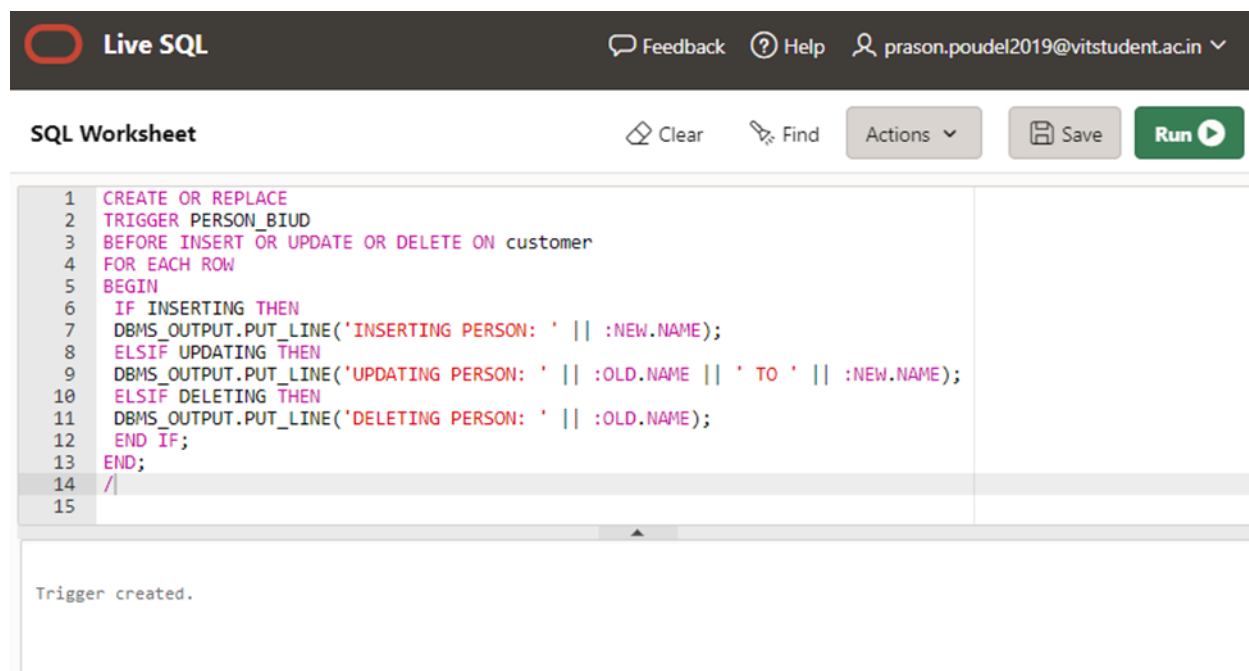
Procedure created.

SQL> set serveroutput on
SQL> begin
2 insertin('C008','MICKEY',98111111,'DUHABI');
3 dbms_output.put_line('inserted successfully');
4 end;
5 /

inserted successfully

PL/SQL procedure successfully completed.
SQL> 19BC2520
```

On the right side, there is a user profile for MICKEY KUMAR ROUNIYAR with options like 'Available', 'Set status message', 'Saved', 'Settings', 'Zoom', 'Keyboard shortcuts', 'About', 'Check for updates', 'Download the mobile app', and 'Sign out'.

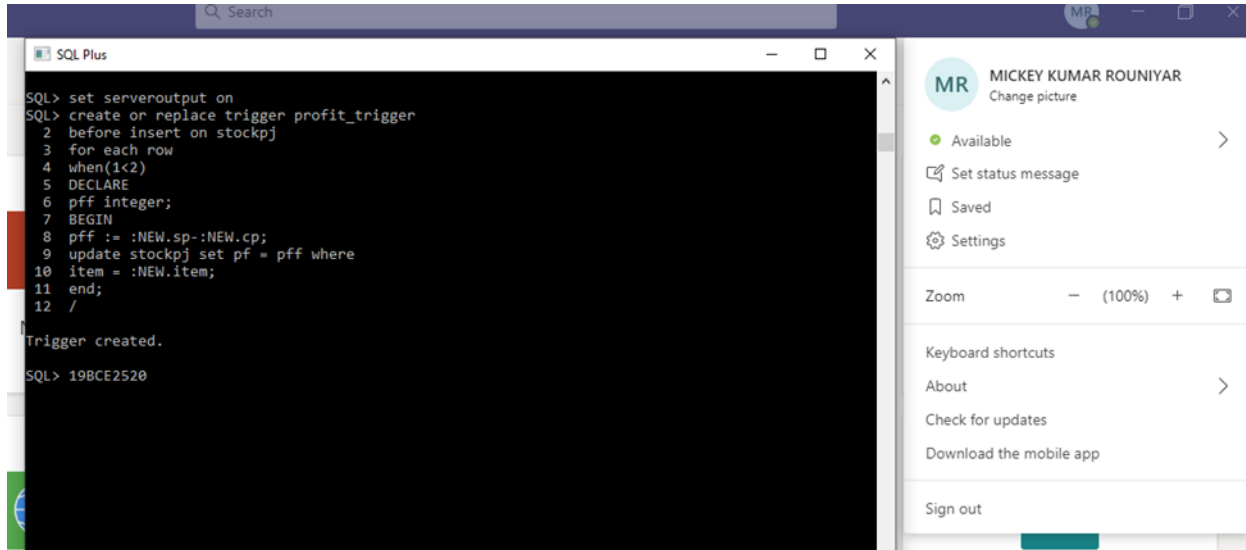


The screenshot shows the Live SQL interface with a SQL Worksheet. The top bar includes 'Live SQL', 'Feedback', 'Help', and a user profile for prason.poudel2019@vitstudent.ac.in. The worksheet contains the following SQL code:

```
1 CREATE OR REPLACE
2 TRIGGER PERSON_BIUD
3 BEFORE INSERT OR UPDATE OR DELETE ON customer
4 FOR EACH ROW
5 BEGIN
6 IF INSERTING THEN
7 DBMS_OUTPUT.PUT_LINE('INSERTING PERSON: ' || :NEW.NAME);
8 ELSIF UPDATING THEN
9 DBMS_OUTPUT.PUT_LINE('UPDATING PERSON: ' || :OLD.NAME || ' TO ' || :NEW.NAME);
10 ELSIF DELETING THEN
11 DBMS_OUTPUT.PUT_LINE('DELETING PERSON: ' || :OLD.NAME);
12 END IF;
13 END;
14 /
15
```

Below the code, the output shows 'Trigger created.'.

Write a Trigger to find and fill the profit of a stock whenever a stock record is inserted into table.



The screenshot shows the SQL Plus command-line interface. The user has entered a series of SQL commands to create a trigger named 'profit_trigger' that fires before an insert on the 'stockpj' table. The trigger calculates a profit 'pff' based on a condition 'when(1<2)' and updates the 'pf' column of the 'stockpj' table. The output shows 'Trigger created.' and a SQL prompt 'SQL> 19BCE2520'. On the right, a sidebar displays the user profile for 'MICKEY KUMAR ROUNIYAR' with options like 'Available', 'Set status message', 'Saved', 'Settings', 'Zoom', 'Keyboard shortcuts', 'About', 'Check for updates', 'Download the mobile app', and 'Sign out'.

```
SQL> set serveroutput on
SQL> create or replace trigger profit_trigger
2 before insert on stockpj
3 for each row
4 when(1<2)
5 DECLARE
6 pff integer;
7 BEGIN
8 pff := :NEW.sp-:NEW.cp;
9 update stockpj set pf = pff where
10 item = :NEW.item;
11 end;
12 /

Trigger created.

SQL> 19BCE2520
```



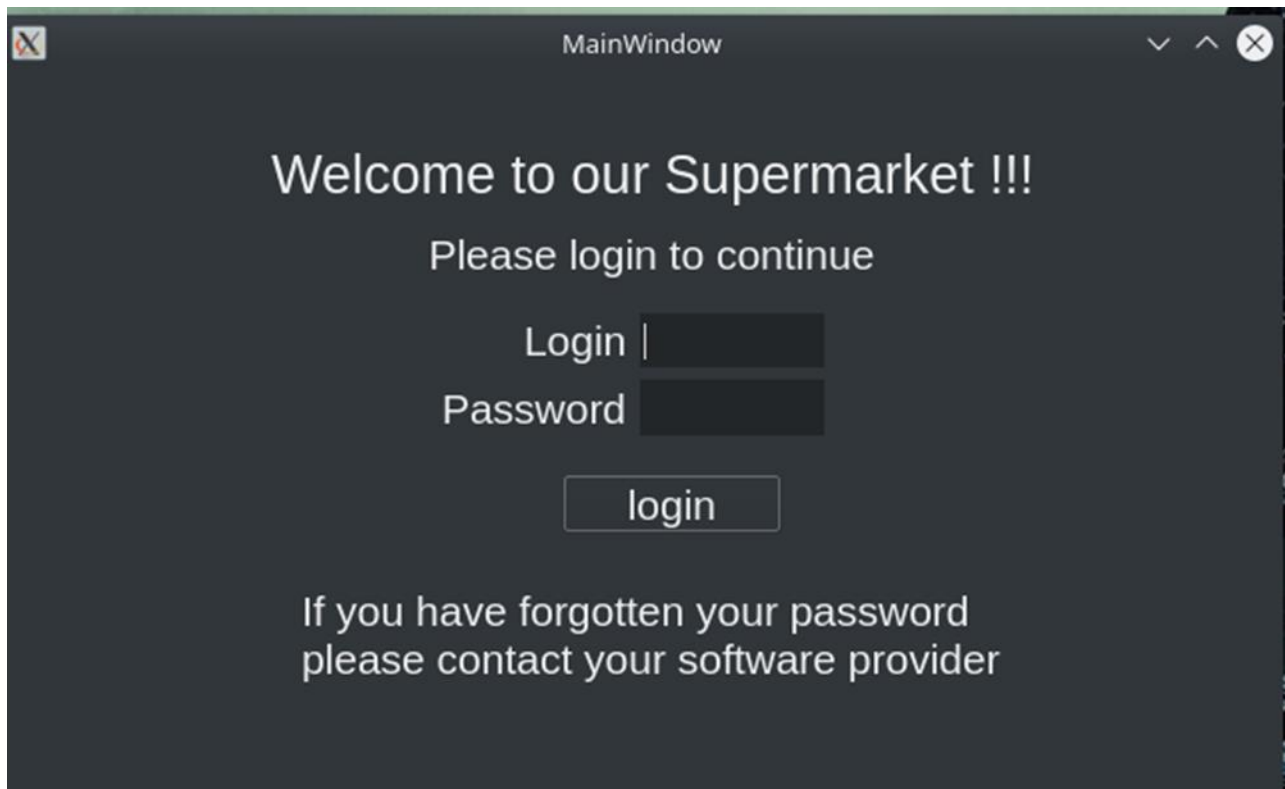
The screenshot shows the 'Live SQL' web interface. The 'SQL Worksheet' tab is active, displaying a PL/SQL function named 'nums' that returns a number. The function logic includes a loop to calculate a total count of employees with a salary greater than 36880. The output pane shows the message 'Function created.' followed by 'Statement processed.' and the result 'The Employees are 3'.

```
1 create or replace function nums
2 return number is
3 total number(2) :=0;
4 begin
5 select count (*) into total
6 from employee where salary>36880;
7 return total;
8 END;
9 /
10 Declare
11 a number(2);
12 begin
13 a :=nums();
14 dbms_output.put_line('The Employees are: ' || a);
15 end;
16 /
17
```

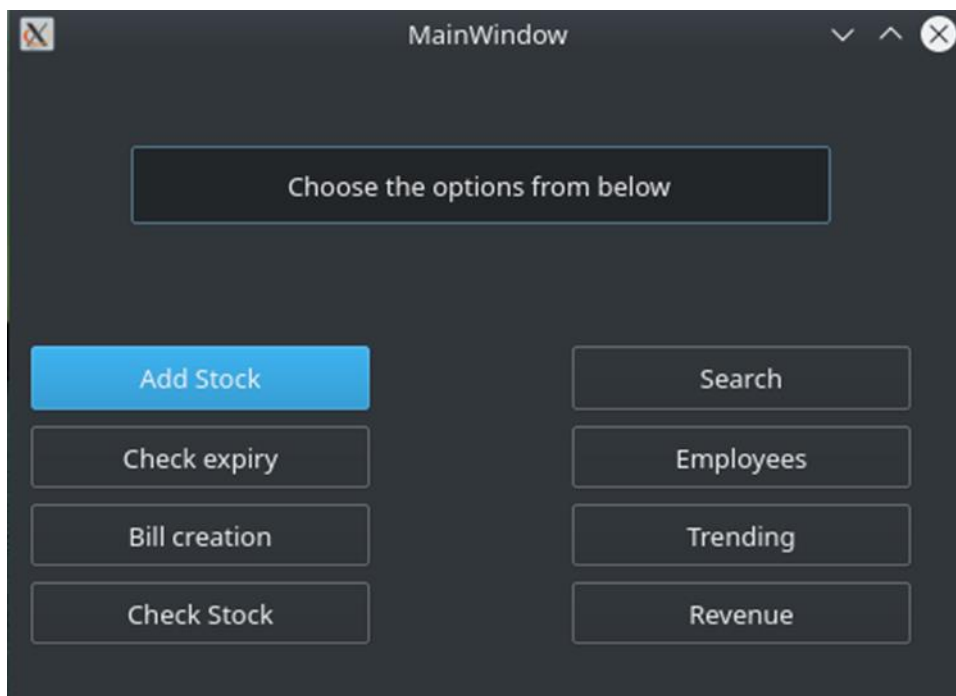
Function created.
Statement processed.
The Employees are 3

6. Screenshot (front end-with explanation)

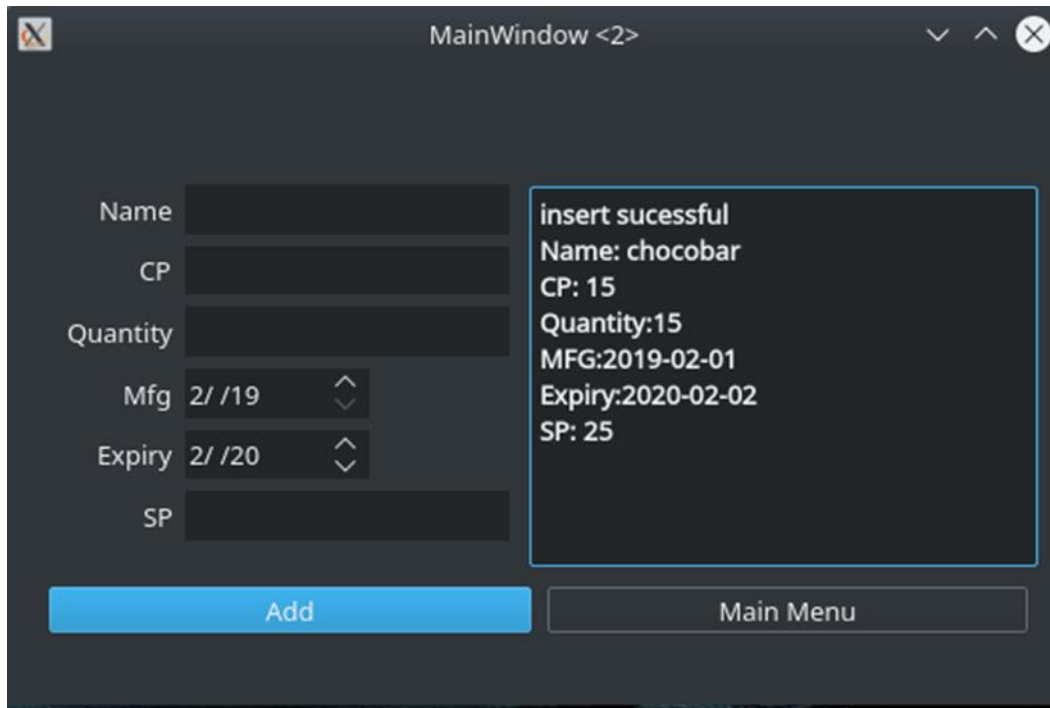
Login Page



Home Page

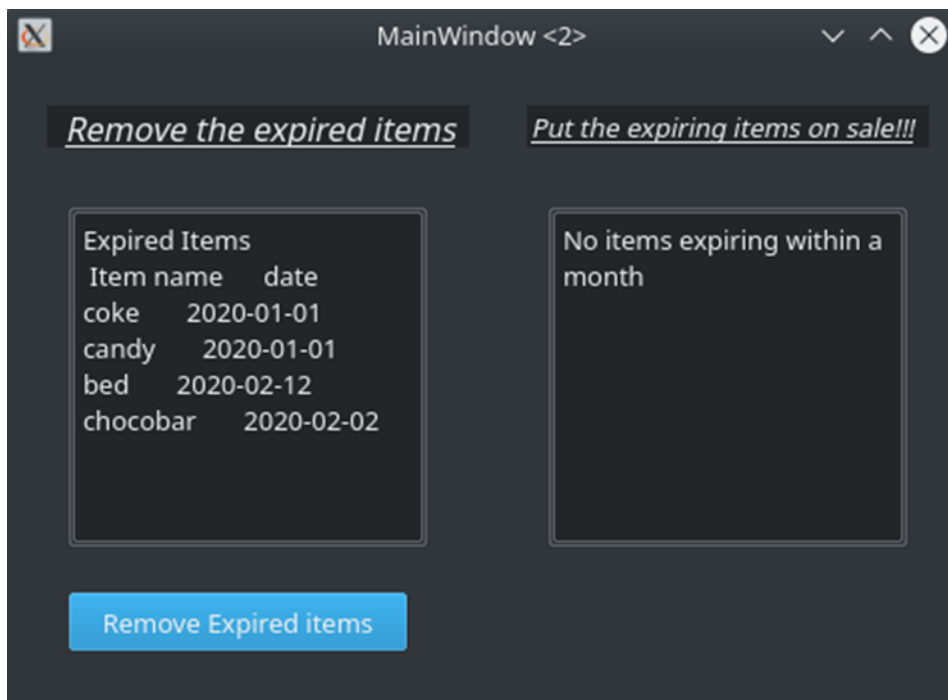


Add Stock



The screenshot shows a window titled "MainWindow <2>". On the left, there are input fields for "Name", "CP", "Quantity", "Mfg" (with a date picker set to 2/ /19), "Expiry" (with a date picker set to 2/ /20), and "SP". A blue "Add" button is at the bottom left. On the right, a text box displays the following information: "insert sucessful", "Name: chocobar", "CP: 15", "Quantity:15", "MFG:2019-02-01", "Expiry:2020-02-02", and "SP: 25". A "Main Menu" button is at the bottom right.

Check Expiry



The screenshot shows a window titled "MainWindow <2>". At the top, there are two tabs: "Remove the expired items" and "Put the expiring items on sale!!!". The "Remove the expired items" tab is active. Below it, a list titled "Expired Items" shows a table with two columns: "Item name" and "date". The table contains the following data:

Item name	date
coke	2020-01-01
candy	2020-01-01
bed	2020-02-12
chocobar	2020-02-02

Below the table is a blue button labeled "Remove Expired items". To the right of the "Expired Items" list, there is a text box that says "No items expiring within a month".

Bill Creation

MainWindow <2>

Billing

Customer name: mickey

Address: duhabi

Phone no.: 981234567

id: 10112

Select item: ball
bat
bed
candy
chocolate

Quantity: 2

Add selected item to bill

Bill for next customer

Items in the bill

ball	2
------	---

total without gst : 40
Total with gst :44.0

Check Stock

MainWindow <2>

STOCK KEEPER

item name	Quantity
coke	21
sprite	40
candy	784
ball	177
bat	27
chau chau	7
fanta	360
ramen	200
bed	8
chocobar	20

Search

MainWindow <2>

Item Customer

Search for an item by any of the options

☒ Name

☐ CP

☐ SP

Results

Product details:

Name: chau chau

Cp: 15

Sp: 25

Quantity: 7

MFG: 2020-01-01

Expiry: 2022-01-01

MainWindow <2>

Item Customer

Search for an customer

☒ Name

☐ Shopped date

☐ Id

results

Customer details

Name: mickey

Address: dubai

Phone: 980123456

Items bought:

- candy X 10
- bed X 2

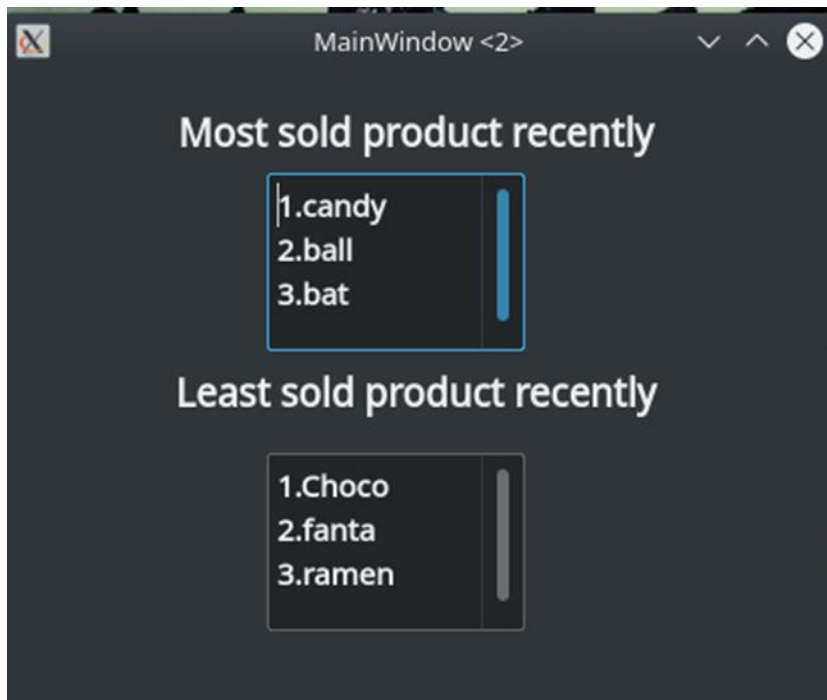
Date of store visit:

Employees

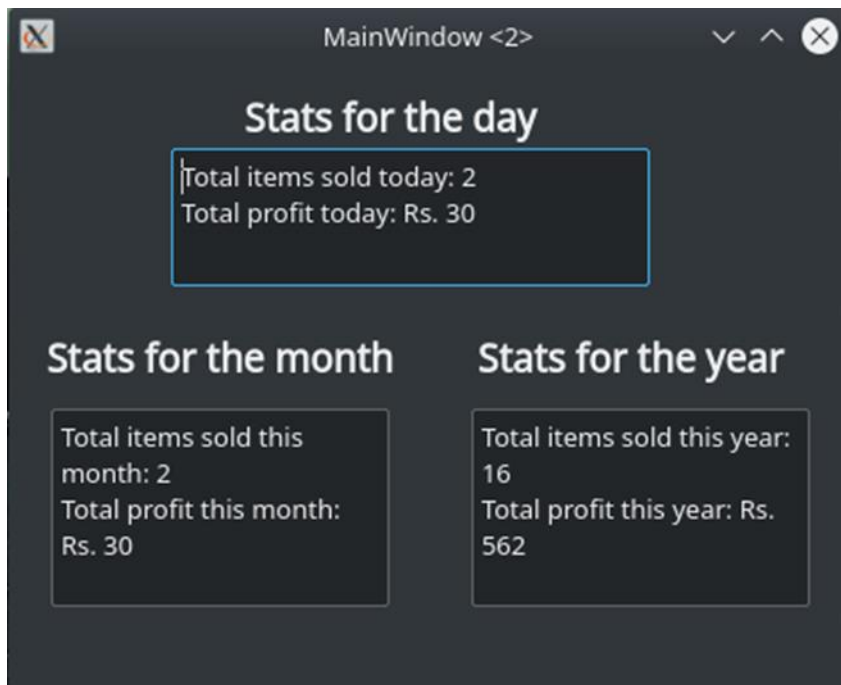
The screenshot shows the 'Enter' tab of a window titled 'MainWindow <2>'. The interface includes a vertical list of labels on the left: 'Id', 'Name', 'Address', 'Post', and 'Salary', each followed by a text input field. Below these fields is a button labeled 'Enter details'. To the right of the input fields is a large, empty rectangular box with a blue border, likely intended for displaying a list or details.

The screenshot shows the 'Search' tab of the same 'MainWindow <2>' window. On the left, there are two radio buttons. The first is checked and labeled 'Name', followed by a text input field. The second is unchecked and labeled 'Id', also followed by a text input field. Below these is a blue button labeled 'Search'. On the right, a section titled 'Results' contains a scrollable list box. The list box displays the following text: 'Employee Information', 'Id:07', 'Name:james', 'Address:baker st.', 'Post:cleaner', and 'Salary: Rs.'.

Trending



Revenue



7. Conclusion and Future Work

7.1 Conclusion

The expected outcome of this project was to develop a software system which would be able to aid the functioning of simple local shops/small businesses by helping them to track the inventory, organize the products, determine the state of the business (Profit and Loss, Most sold items, etc.) and the generation of bill during the sale and we were successfully able to achieve the expected results and thus present our fully functioning SUPERMARKET MANAGEMENT SOFTWARE.

Further Improvements and Updates to the software will be made along with time.

7.2 Future Work

We want to integrate the following features in the future:

- Discount on products will be generated according to their sale reports.(i.e. The least sold items will discounted accordingly to improve the sales)
- Coupons / Promo Code application
- Support to handheld devices (IOS, Android, etc.)

References:

- [1] R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7 th Edition, 2015
- [2] Thomas Connolly, Carolyn Begg, Database Systems: A l Practical Approach to Design, Implementation and Managementll, 6th Edition, 2012
- [3] Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming by Mark Summerfield
- [4] MySQL for Python by Albert Lukaszewski
- [5] Moskowitz,Robert,"Using Your Computer for Inventory Control", Accvision Retrieved August 17, 2010.
- [6] "Inventory Management Software". EGA Futura. Retrieved 23 November 2012.
- [7] Lockard, Robert (29 November 2010) "3 Advantages of Using Inventory Management Software". Inventory System Software Blog. Retrieved 23 November 2012.
- [8]"Inventory Management Organism" by A Rohini, Wubshetasamnew Woldeyohannshiluf and Teshometeklemicheal Walelignayimelo
- [9] "Tracking Inventory" by Lesonsky, Rieva (1998) Entrepreneur Magazine

Google Drive link [all the code are available in the link]

If you want to see the code then you can click to the given google drive link which is attached below:

<https://drive.google.com/drive/folders/1UF9qIUg2z0Vg3GuUX9ZynjXQN2iQGBM5?usp=sharing>