

PES UNIVERSITY
Department of Computer Science & Engineering



DBMS - UE20CS301
Mini Project
Inventory Management System

Submitted to:

Dr. Geetha D
Associate Professor

Submitted By:

Name: Vishwa Mehul Mehta
SRN: PES2UG20CS389
Semester: V
Section: F

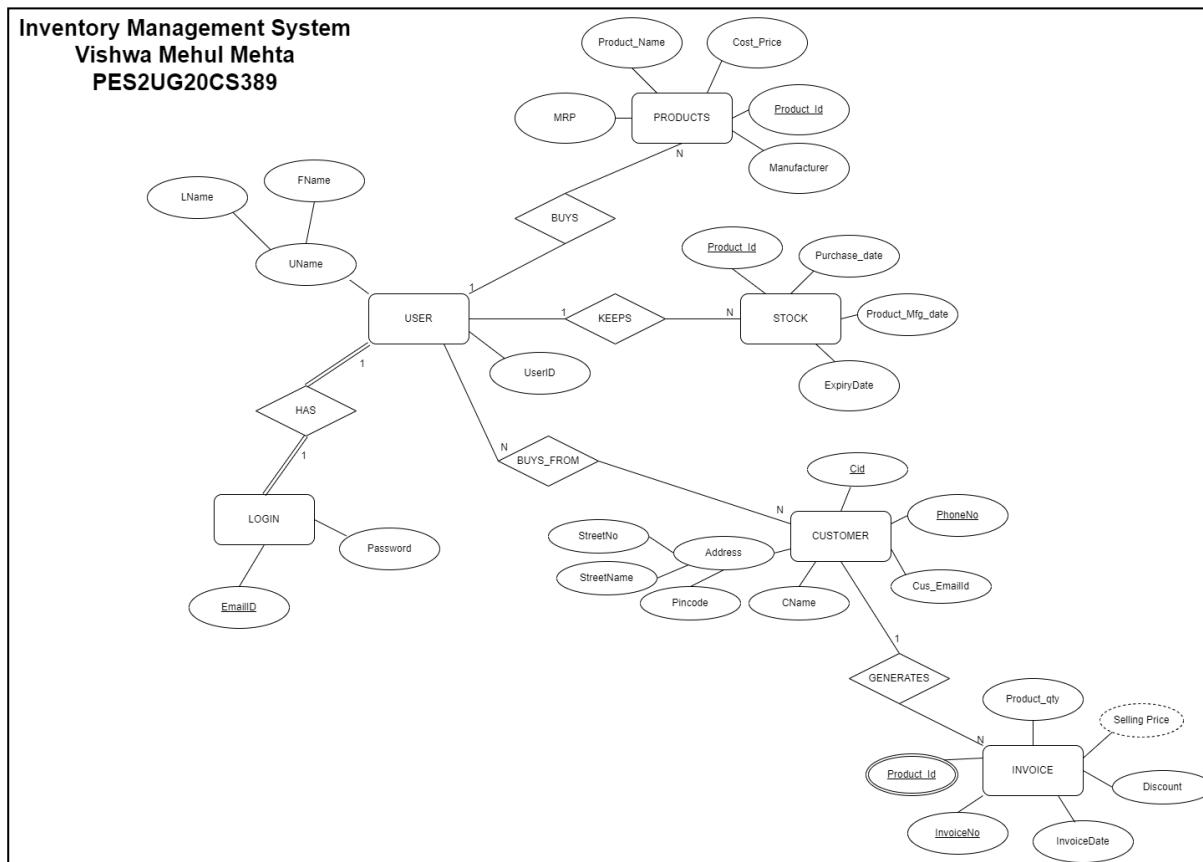
Table of Contents

Sl.No	Title	Page No
1.	Short Description and Scope of the Project	3
2.	Entity Relationship Diagram	4
3.	Relational Schema	5
4.	DDL Statements-Building the Database	6
5.	Populating the Database	9
6.	Join Queries	13
7.	Attribute Functions	15
8.	Set Operations	17
9.	Functions and Procedures	20
10.	Triggers and Cursors	24
11.	Frontend Application	30
12.	Conclusion	58
13.	References	59

1. Short Description and Scope of the Project

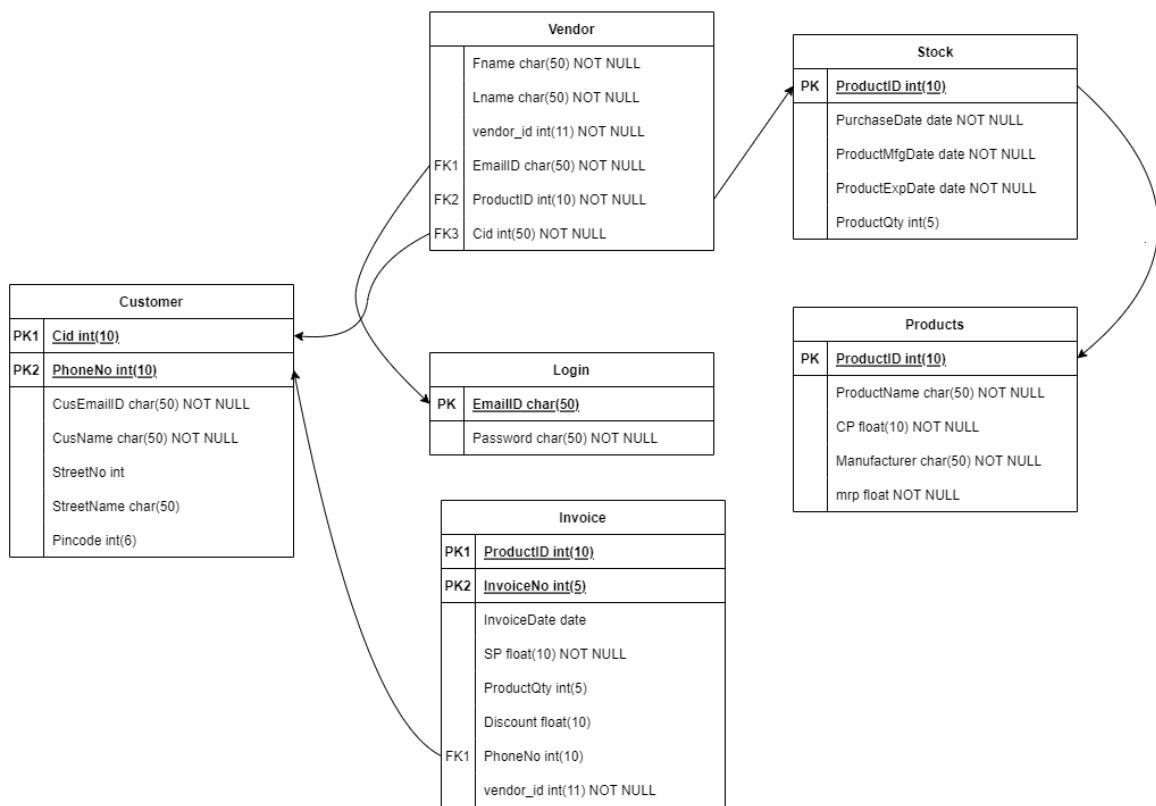
ListAll is a web application that monitors and manages the goods and products in stock. The application uses a database and database operations to store information about the retailers and their respective products that are sold. It displays a list of all products and their quantities, customers who bought the items and their details along with invoice for each product purchased. It allows for performing CRUD operations for all the details. ListAll uses a simple frontend using Python based Streamlit and MySQL for the backend. The application aims to find profit/loss and generate it for a particular month or year as a future application.

2. ER Diagram



3. Relational Schema

Inventory Management System
Vishwa Mehul Mehta
PES2UG20CS389



4. DDL statements - Building the database

```
-- Table structure for table `customer`  
--  
CREATE TABLE IF NOT EXISTS `customer` (  
  `cust_id` int(10) NOT NULL,  
  `phone_no` bigint(10) NOT NULL,  
  `cust_email_id` varchar(50) NOT NULL,  
  `cust_name` varchar(50) NOT NULL,  
  `str_no` int(11) DEFAULT NULL,  
  `str_name` varchar(50) DEFAULT NULL,  
  `pincode` int(6) DEFAULT NULL,  
  PRIMARY KEY (`cust_id`, `phone_no`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;  
  
-- Table structure for table `invoice`  
--  
CREATE TABLE IF NOT EXISTS `invoice` (  
  `prod_id` int(10) NOT NULL,  
  `invoice_no` int(10) NOT NULL,  
  `invoice_date` date NOT NULL,  
  `selling_price` float NOT NULL,  
  `prod_qty` int(11) DEFAULT '0',  
  `discount` float DEFAULT NULL,  
  `phone_no` bigint(10) DEFAULT NULL,  
  `vendor_id` int(11) NOT NULL,  
  PRIMARY KEY (`prod_id`, `invoice_no`),  
  KEY `phone_no` (`phone_no`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
-- Table structure for table `login`
-- 

CREATE TABLE IF NOT EXISTS `login` (
  `email_id` varchar(50) NOT NULL,
  `pass` varchar(50) NOT NULL,
  PRIMARY KEY (`email_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Table structure for table `product`
-- 

CREATE TABLE IF NOT EXISTS `product` (
  `product_id` int(10) NOT NULL,
  `product_name` varchar(50) NOT NULL,
  `cost_price` float NOT NULL,
  `manufacturer` varchar(50) NOT NULL,
  `mrp` float NOT NULL,
  PRIMARY KEY (`product_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Table structure for table `stock`
-- 

CREATE TABLE IF NOT EXISTS `stock` (
  `pid` int(10) NOT NULL,
  `purchase_date` date NOT NULL,
  `mfg_date` date NOT NULL,
  `pqty` int(11) DEFAULT '0',
  PRIMARY KEY (`pid`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Table structure for table `vendor`
-- 
```

```
CREATE TABLE IF NOT EXISTS `vendor` (
  `fname` varchar(50) NOT NULL,
  `lname` varchar(50) NOT NULL,
  `vendor_id` int(11) NOT NULL,
  `email_id` varchar(50) NOT NULL,
  `product_id` int(10) DEFAULT NULL,
  `cust_id` int(10) DEFAULT NULL,
  KEY `email_id` (`email_id`),
  KEY `cust_id` (`cust_id`),
  KEY `product_id` (`product_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

5. Populating the Database

```
-- Dumping data for table `customer`  
--  
INSERT INTO `customer` (`cust_id`, `phone_no`, `cust_email_id`,  
`cust_name`, `str_no`, `str_name`, `pincode`) VALUES  
(57317, 5289998172, 'inderpal@gmail.com', 'Inderpal Ravinder', 12,  
'Goel Street', 649721),  
(26864, 8043894922, 'ishi@gmail.com', 'Ishita Devraj', 14,  
'Ramkissoon Street', 427425),  
(58462, 8545588678, 'anikp@gmail.com', 'Anik Prabhakar', 56,  
'Sekhon Street', 139845),  
(63698, 3816889497, 'ninadram@gmail.com', 'Ninad Ram', 37, 'Kari  
Street', 456639),  
(33295, 9489333654, 'devis@yahoo.com', 'Devi Sunder', 28,  
'Subramaniam Street', 376921),  
(22150, 9056364822, 'rama123@yahoo.com', 'Ramadevi Gopala', 57,  
'Sen Street', 758086),  
(76443, 7294778333, 'pravinajaya@gmail.com', 'Pravina Jaya', 22,  
'Raja Street', 591648),  
(45406, 4305436406, 'anilabhi@gmail.com', 'Anil Abhishek', 71,  
'Shroff Street', 234740),  
(26864, 8145956923, 'nilofara@hotmail.com', 'Nilofar Amardeep', 83,  
'Vig Street', 158657),  
(43143, 5636459498, 'vinayarvind24@hotmail.com', 'Vinay Aravind',  
26, 'Chakraborty Street', 525343);  
  
-- Dumping data for table `invoice`  
--  
INSERT INTO `invoice` (`prod_id`, `invoice_no`, `invoice_date`,  
`selling_price`, `prod_qty`, `discount`, `phone_no`, `vendor_id`) VALUES  
(59854, 38980, '2022-10-22', 22.8, 2, 0.2, 5289998172, 75071),  
(59854, 85957, '2022-10-23', 22.8, 2, 0.2, 8043894922, 75071),  
(59854, 99454, '2022-10-23', 22.8, 2, 0.2, 8545588678, 40101);
```

```
-- Dumping data for table `login`  
--  
INSERT INTO `login` (`email_id`, `pass`) VALUES  
('harry@gmail.com', 'har1234%'),  
('ramesh@gmail.com', 'rameshH234'),  
('ram@gmail.com', 'ram135&'),  
('ranjeeta@gmail.com', 'ran234%'),  
('rakesh@gmail.com', 'rakeshm345');  
  
-- Dumping data for table `product`  
--  
INSERT INTO `product` (`product_id`, `product_name`, `cost_price`,  
`manufacturer`, `mrp`) VALUES  
(59854, 'milk', 15, 'Amul', 19),  
(53028, 'curd', 30, 'Amul', 34),  
(56145, 'butter', 25, 'Amul', 28),  
(81010, 'cow milk', 15, 'Nandini', 17),  
(99016, 'yogurt', 25, 'Nandini', 30),  
(15292, 'paneer', 55, 'Milky Mist', 60),  
(45124, 'cheese', 85, 'Amul', 90),  
(14270, 'ghee', 440, 'Nandini', 450),  
(86934, 'ice cream family pack', 180, 'Quality Walls', 200),  
(98390, 'ice cream cone', 35, 'Cornetto', 40);  
  
-- Dumping data for table `stock`  
--  
INSERT INTO `stock` (`pid`, `purchase_date`, `mfg_date`, `pqty`)  
VALUES  
(59854, '2022-10-21', '2022-10-21', 80),  
(53028, '2022-10-18', '2022-10-18', 90),
```

```

(56145, '2022-10-12', '2022-10-10', 60),
(81010, '2022-10-21', '2022-10-21', 60),
(99016, '2022-10-17', '2022-10-16', 50),
(15292, '2022-10-19', '2022-10-19', 30),
(45124, '2022-10-04', '2022-09-29', 35),
(14270, '2022-10-02', '2022-09-20', 40),
(86934, '2022-09-20', '2022-09-10', 50),
(98390, '2022-10-01', '2022-09-30', 35);

-- Dumping data for table `vendor`

--
INSERT INTO `vendor` (`fname`, `lname`, `vendor_id`, `email_id`,
`product_id`, `cust_id`) VALUES
('harry', 'singh', 75071, 'harry@gmail.com', 59854, 57317),
('harry', 'singh', 75071, 'harry@gmail.com', 59854, 26864),
('harry', 'singh', 75071, 'harry@gmail.com', 59854, 58462),
('harry', 'singh', 75071, 'harry@gmail.com', 53028, 57317),
('harry', 'singh', 75071, 'harry@gmail.com', 14270, 58462),
('harry', 'singh', 75071, 'harry@gmail.com', 59854, 63698),
('harry', 'singh', 75071, 'harry@gmail.com', 98390, 63698),
('ramesh', 'sharma', 40101, 'ramesh@gmail.com', 59854, 33295),
('ramesh', 'sharma', 40101, 'ramesh@gmail.com', 15292, 33295),
('ramesh', 'sharma', 40101, 'ramesh@gmail.com', 59854, 33295),
('ramesh', 'sharma', 40101, 'ramesh@gmail.com', 14270, 22150),
('ramesh', 'sharma', 40101, 'ramesh@gmail.com', 59854, 22150),
('ram', 'c', 70978, 'ram@gmail.com', 53028, 76443),
('ram', 'c', 70978, 'ram@gmail.com', 59854, 76443),
('ranjeeta', 's', 94138, 'ranjeeta@gmail.com', 59854, 45406),
('ranjeeta', 's', 94138, 'ranjeeta@gmail.com', 45124, 26864),
('ranjeeta', 's', 94138, 'ranjeeta@gmail.com', 98390, 45406),
('rakesh', 'sharma', 29690, 'rakesh@gmail.com', 81010, 43143),

```

```
('rakesh', 'sharma', 29690, 'rakesh@gmail.com', 15292, 43143);
```

6. Join Queries

- a. List of all customers that have an invoice associated with them.

Query:

```
SELECT customer.cust_id, customer.cust_name, invoice.invoice_no
FROM customer INNER JOIN invoice WHERE customer.phone_no = invoice.phone_no;
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL:3306
- Database:** inventory_management_system
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Show query box.
- Message Bar:** Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)
- Query Editor:** SELECT customer.cust_id, customer.cust_name, invoice.invoice_no FROM customer INNER JOIN invoice WHERE customer.phone_no = invoice.phone_no;
- Table View:** cust_id, cust_name, invoice_no
- Data:**

cust_id	cust_name	invoice_no
57317	Inderpal Ravinder	38980
26864	Ishita Devraj	85957
58462	Anik Prabhakar	99454

- b. List all the cost prices of products in stock.

Query:

```
SELECT stock.pid, product.product_name, product.cost_price
FROM product INNER JOIN stock WHERE product.product_id = stock.pid;
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL:3306
- Database:** inventory_management_system
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Show query box.
- Message Bar:** Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.)
- Query Editor:** SELECT stock.pid, product.product_name, product.cost_price FROM product INNER JOIN stock WHERE product.product_id = stock.pid;
- Table View:** pid, product_name, cost_price
- Data:**

pid	product_name	cost_price
14270	ghee	450
15292	paneer	60
45124	cheese	90
53028	curd	34
56145	butter	28
59854	milk	19
81010	cow milk	17
86934	ice cream family pack	200
98390	ice cream cone	40
99016	yogurt	30

c. List all customer names who visit a certain vendor.

Query:

```
SELECT customer.cust_name, vendor.vendor_id FROM vendor INNER JOIN
customer WHERE customer.cust_id = vendor.cust_id AND vendor.vendor_
id = 75071;
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL 3306
- Database:** inventory_management_system
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, More
- Query Result:** A green bar at the top indicates "Showing rows 0 - 6 (7 total, Query took 0.0005 seconds.)". The query executed was:

```
SELECT customer.cust_name, vendor.vendor_id FROM vendor INNER JOIN customer WHERE customer.cust_id = vendor.cust_id AND vendor.vendor_id = 75071;
```
- Table Data:** A table titled "+ Options" showing the results of the query:

cust_name	vendor_id
Inderpal Ravinder	75071
Ishita Devraj	75071
Anik Prabhakar	75071
Inderpal Ravinder	75071
Anik Prabhakar	75071
Ninad Ram	75071
Ninad Ram	75071

d. List quantity of all products in stock along with product details.

Query:

```
SELECT product.product_name, stock.pqty FROM product INNER JOIN stock
WHERE product.product_id = stock.pid;
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL 3306
- Database:** inventory_management_system
- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, More
- Query Result:** A green bar at the top indicates "Showing rows 0 - 9 (10 total, Query took 0.0018 seconds.)". The query executed was:

```
SELECT product.product_name, stock.pqty FROM product INNER JOIN stock WHERE product.product_id = stock.pid;
```
- Table Data:** A table titled "+ Options" showing the results of the query:

product_name	pqty
milk	80
curd	90
butter	60
cow milk	60
yogurt	50
paneer	30
cheese	35
ghee	40
ice cream family pack	50
ice cream cone	35

7. Aggregate Functions:

- a. The total number of products each vendor sold.

Query:

```
SELECT vendor.vendor_id, vendor.fname, vendor.lname, COUNT(DISTINCT vendor.product_id) FROM vendor GROUP BY vendor.vendor_id;
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: inventory_management_system
- Table: vendor
- Toolbar buttons: Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Triggers.
- A warning message: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." with a help icon.
- Message bar: "Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)"
- SQL pane:

```
SELECT vendor.vendor_id, vendor.fname, vendor.lname, COUNT(DISTINCT vendor.product_id) FROM vendor GROUP BY vendor.vendor_id;
```
- Buttons below SQL pane: Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh].
- Table pane:

vendor_id	fname	lname	COUNT(DISTINCT vendor.product_id)
29690	rakesh	sharma	2
40101	ramesh	sharma	3
70978	ram	c	2
75071	harry	singh	4
94138	ranjeeta	s	3
- Options: Show all | Number of rows: 25 | Filter rows: Search this table.

- b. The number of customers who bought milk.

Query:

```
SELECT COUNT(DISTINCT vendor.cust_id) CountOfMilk FROM vendor INNER JOIN product WHERE product.product_name LIKE "%milk%";
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: inventory_management_system
- Table: vendor
- Toolbar buttons: Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Triggers.
- A warning message: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." with a help icon.
- Message bar: "Your SQL query has been executed successfully."
- SQL pane:

```
SELECT COUNT(DISTINCT vendor.cust_id) CountOfMilk FROM vendor INNER JOIN product WHERE product.product_name LIKE "%milk%";
```
- Buttons below SQL pane: Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh].
- Table pane:

CountOfMilk
9

- c. The vendor who sells to maximum number of customers.

Query:

```

SELECT vendor.vendor_id, COUNT(DISTINCT vendor.cust_id) AS MaximumCustomers
FROM vendor GROUP BY vendor.vendor_id HAVING COUNT(DISTINCT vendor.cust_id) =
(SELECT MAX(Customers.MaximumCustomers) FROM (SELECT vendor.vendor_id,
COUNT(DISTINCT vendor.cust_id) as MaximumCustomers FROM vendor
GROUP BY vendor.vendor_id) AS Customers);

```

Screenshot:

Your SQL query has been executed successfully.

```

SELECT vendor.vendor_id, COUNT(DISTINCT vendor.cust_id) AS MaximumCustomers
FROM vendor GROUP BY vendor.vendor_id HAVING COUNT(DISTINCT vendor.cust_id) =
(SELECT MAX(Customers.MaximumCustomers) FROM (SELECT vendor.vendor_id, COUNT(DISTINCT vendor.cust_id) as MaximumCustomers
FROM vendor GROUP BY vendor.vendor_id) AS Customers);

```

+ Options	
vendor_id	MaximumCustomers
75071	4

d. Average cost price of all Amul products.

Query:

```

SELECT AVG(product.cost_price) AS AverageCostPrice,
product.manufacturer FROM product WHERE product.manufacturer =
"Amul" GROUP BY product.manufacturer;

```

Screenshot:

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```

SELECT AVG(product.cost_price) as AverageCostPrice, product.manufacturer FROM product WHERE product.manufacturer = "Amul" GROUP BY product.manufacturer;

```

+ Options	
AverageCostPrice	manufacturer
42.75	Amul

8. Set Operations:

- a. Common products sold by vendor 75071 and 40101.

Query:

```
SELECT DISTINCT vendor.product_id FROM vendor WHERE
vendor.vendor_id = 75071 AND vendor.product_id IN (SELECT
vendor.product_id FROM vendor WHERE vendor.vendor_id = 40101);
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: inventory_management_system
- Table: vendor
- Toolbar buttons: Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations.
- Message bar: Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Status bar: Showing rows 0 - 1 (2 total, Query took 0.0009 seconds.)
- SQL pane: SELECT DISTINCT vendor.product_id FROM vendor WHERE vendor.vendor_id = 75071 AND vendor.product_id IN (SELECT vendor.product_id FROM vendor WHERE vendor.vendor_id = 40101);
- Buttons below SQL pane: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh.
- Table toolbar: Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None.
- Table data:
 - + Options
 - product_id
 - 59854
 - 14270

- b. Products sold by Amul or Milky Mist.

Query:

```
SELECT product.product_id, product.product_name,
product.manufacturer FROM product WHERE product.manufacturer =
"Amul" UNION SELECT product.product_id, product.product_name,
product.manufacturer FROM product WHERE product.manufacturer =
"Milky Mist";
```

Screenshot:

Server: MySQL:3306 » Database: inventory_management_system » Table: product

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)

```
SELECT product.product_id, product.product_name, product.manufacturer FROM product WHERE product.manufacturer = "Amul" UNION SELECT product.product_id, product.product_name, product.manufacturer FROM product WHERE product.manufacturer = "Milky Mist";
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

product_id	product_name	manufacturer
59854	milk	Amul
53028	curd	Amul
56145	butter	Amul
45124	cheese	Amul
15292	paneer	Milky Mist

c. Products not sold by Nandini.

Query:

```
SELECT product.product_id, product.product_name,
product.manufacturer FROM product WHERE product.manufacturer NOT IN
(SELECT product.manufacturer FROM product WHERE
product.manufacturer = "Nandini");
```

Screenshot:

Server: MySQL:3306 » Database: inventory_management_system » Table: product

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Showing rows 0 - 6 (7 total, Query took 0.0022 seconds.)

```
SELECT product.product_id, product.product_name, product.manufacturer FROM product WHERE product.manufacturer NOT IN (SELECT
product.manufacturer FROM product WHERE product.manufacturer = "Nandini");
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

product_id	product_name	manufacturer
59854	milk	Amul
53028	curd	Amul
56145	butter	Amul
15292	paneer	Milky Mist
45124	cheese	Amul
86934	ice cream family pack	Quality Walls
98390	ice cream cone	Cornetto

d. Vendors who don't have any sold products in invoice.

Query:

```
SELECT DISTINCT vendor.fname, vendor.lname, vendor.vendor_id FROM vendor WHERE vendor.vendor_id NOT IN (SELECT invoice.vendor_id FROM invoice);
```

Screenshot:

The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL-3306
- Database:** inventory_management_system
- Table:** vendor
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Triggers.
- Status Bar:** Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Message Bar:** Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)
- SQL Editor:** SELECT DISTINCT vendor.fname, vendor.lname, vendor.vendor_id FROM vendor WHERE vendor.vendor_id NOT IN (SELECT invoice.vendor_id FROM invoice);
- Buttons:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Filter:** Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
- Data Table:** fname | lname | vendor_id
ram | c | 70978
ranjeeta | s | 94138
rakesh | sharma | 29690

9. Functions and Procedures:

a. Function which shows which stocks need to be re-filled ie if the stock quantity is less than 50

Function creation:

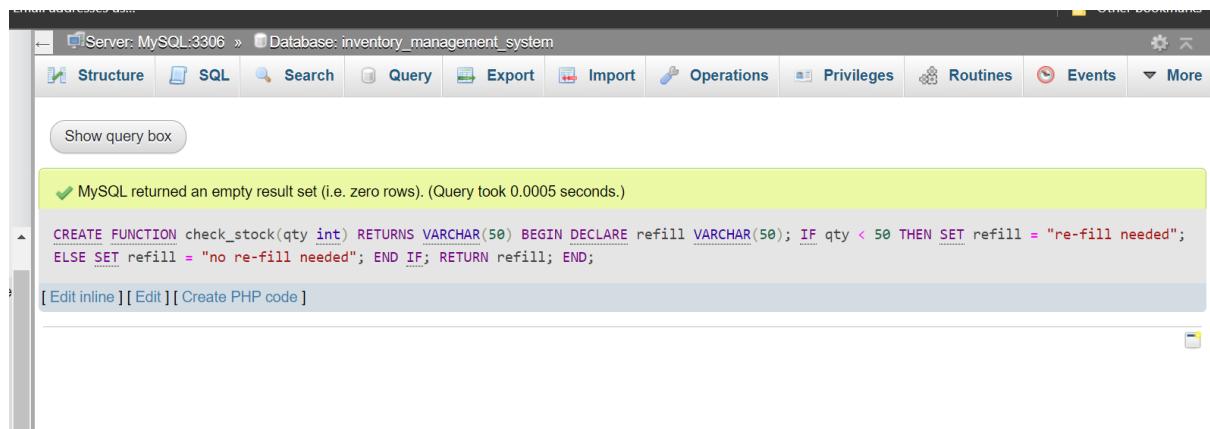
```
DELIMITER $  
CREATE FUNCTION check_stock(qty int)  
RETURNS VARCHAR(50)  
BEGIN  
    DECLARE refill VARCHAR(50);  
    IF qty < 50 THEN  
        SET refill = "re-fill needed";  
    ELSE  
        SET refill = "no re-fill needed";  
    END IF;  
    RETURN refill;  
END $
```

DELIMITER ;

Function call:

```
SELECT stock.pid, stock.pqty, check_stock(stock.pqty) FROM stock;
```

Screenshots:



Server: MySQL:3306 » Database: inventory_management_system » Table: stock

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#)

Showing rows 0 - 9 (10 total, Query took 0.0015 seconds.)

```
SELECT stock.pid, stock.pqty, check_stock(stock.pqty) FROM stock;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	pid	pqty	check_stock(stock.pqty)
<input type="checkbox"/>	59854	80	no re-fill needed
<input type="checkbox"/>	53028	90	no re-fill needed
<input type="checkbox"/>	56145	60	no re-fill needed
<input type="checkbox"/>	81010	60	no re-fill needed
<input type="checkbox"/>	99016	50	no re-fill needed
<input type="checkbox"/>	15292	30	re-fill needed
<input type="checkbox"/>	45124	35	re-fill needed
<input type="checkbox"/>	14270	40	re-fill needed
<input type="checkbox"/>	86934	50	no re-fill needed
<input type="checkbox"/>	98390	35	re-fill needed

[Console](#)

b. Procedure to update the product quantity to given value and purchase date to current date if the quantity is less than 50.

Procedure creation:

```
DELIMITER $  
CREATE PROCEDURE update_qty(IN qty int)  
BEGIN  
    UPDATE stock SET stock.pqty = qty, stock.purchase_date =  
    CURRENT_DATE WHERE stock.pqty < 50;  
END $  
DELIMITER ;
```

Procedure call:

```
CALL update_qty(50);
```

Screenshots:

This screenshot shows the MySQL Workbench interface. In the top navigation bar, it says "Server: MySQL:3306" and "Database: inventory_management_system". Below the navigation bar is a toolbar with tabs: Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, and More. A "Show query box" button is also present. The main area displays a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0019 seconds.)" followed by the SQL code for creating a stored procedure named "update_qty". The code is as follows:

```
CREATE PROCEDURE update_qty(IN qty int) BEGIN UPDATE stock SET stock.pqty = qty, stock.purchase_date = CURRENT_DATE WHERE stock.pqty < 50; END;
```

Below the code are three buttons: [Edit inline], [Edit], and [Create PHP code].

This screenshot shows the MySQL Workbench interface with the "Table: stock" selected. The top navigation bar and toolbar are identical to the previous screenshot. The main area displays the "stock" table data. The table has columns: pid, purchase_date, mfg_date, and pqty. There are 10 rows of data. The last row is currently selected. At the bottom of the table area, there are buttons for "Check all", "With selected:", "Edit", "Copy", "Delete", and "Export".

	pid	purchase_date	mfg_date	pqty
<input type="checkbox"/>	59854	2022-10-21	2022-10-21	80
<input type="checkbox"/>	53028	2022-10-18	2022-10-18	90
<input type="checkbox"/>	56145	2022-10-12	2022-10-10	60
<input type="checkbox"/>	81010	2022-10-21	2022-10-21	60
<input type="checkbox"/>	99016	2022-10-17	2022-10-16	50
<input type="checkbox"/>	15292	2022-10-19	2022-10-19	30
<input type="checkbox"/>	45124	2022-10-04	2022-09-29	35
<input type="checkbox"/>	14270	2022-10-02	2022-09-20	40
<input type="checkbox"/>	86934	2022-09-20	2022-09-10	50
<input type="checkbox"/>	98390	2022-10-01	2022-09-30	35

This screenshot shows the MySQL Workbench interface. The top navigation bar and toolbar are identical to the previous screenshots. The main area displays a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0011 seconds.)" followed by the SQL code for calling the "update_qty" procedure with the argument "50". Below the code are three buttons: [Edit inline], [Edit], and [Create PHP code].

← Server: MySQL:3306 » Database: inventory_management_system » Table: stock

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#)

Show all | Number of rows: 25 Filter rows: Sort by key

+ Options

	pid	purchase_date	mfg_date	pqty
<input type="checkbox"/>	59854	2022-10-21	2022-10-21	80
<input type="checkbox"/>	53028	2022-10-18	2022-10-18	90
<input type="checkbox"/>	56145	2022-10-12	2022-10-10	60
<input type="checkbox"/>	81010	2022-10-21	2022-10-21	60
<input type="checkbox"/>	99016	2022-10-17	2022-10-16	50
<input type="checkbox"/>	15292	2022-11-17	2022-10-19	50
<input type="checkbox"/>	45124	2022-11-17	2022-09-29	50
<input type="checkbox"/>	14270	2022-11-17	2022-09-20	50
<input type="checkbox"/>	86934	2022-09-20	2022-09-10	50
<input type="checkbox"/>	98390	2022-11-17	2022-09-30	50

↑ Check all With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

10. Triggers and Cursors:

a. If manufacturing or purchase date is greater than the current date and manufacturing date should be less than purchase date then show an error.

Trigger creation:

```
DELIMITER $  
  
CREATE TRIGGER on_insert_stock  
BEFORE INSERT  
ON stock FOR EACH ROW  
  
BEGIN  
  
    DECLARE err_msg1 varchar(100);  
    DECLARE err_msg2 varchar(100);  
    DECLARE err_msg3 varchar(100);  
  
    SET err_msg1 = "Mfg/Purchase Date must be before the current  
date";  
  
    SET err_msg2 = "Mfg Date should be before Purchase Date";  
    SET err_msg3 = "Product quantity should be minimum 50";  
  
    IF (new.purchase_date > CURRENT_DATE OR new.mfg_date >  
CURRENT_DATE) THEN  
  
        SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = err_msg1;  
        ELSEIF new.mfg_date > new.purchase_date THEN  
        SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = err_msg2;  
        ELSEIF new.pqty < 50 THEN  
        SIGNAL SQLSTATE '45000'  
            SET MESSAGE_TEXT = err_msg3;  
    END IF;  
  
END $  
DELIMITER ;
```

Incorrect insert:

```

INSERT INTO stock VALUES (86934, '2022-11-01', '2022-12-01', 50);
INSERT INTO stock VALUES (86934, '2022-12-11', '2022-12-01', 50);
INSERT INTO stock VALUES (86934, '2022-10-01', '2022-11-01', 50);
INSERT INTO stock VALUES (86934, '2022-11-10', '2022-11-01', 5);

```

Correct insert:

```
INSERT INTO stock VALUES (86934, '2022-11-01', '2022-11-10', 50);
```

Screenshots:

This screenshot shows the MySQL Workbench interface. The top navigation bar includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, and More. The main area displays a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0175 seconds.)". Below this, there is a code editor containing a trigger definition:

```

CREATE TRIGGER on_insert_stock BEFORE INSERT ON stock FOR EACH ROW BEGIN DECLARE err_msg1 varchar(100); DECLARE err_msg2 varchar(100);  
DECLARE err_msg3 varchar(100); SET err_msg1 = "Mfg/Purchase Date must be before the current date"; SET err_msg2 = "Mfg Date should be  
before Purchase Date"; SET err_msg3 = "Product quantity should be minimum 50"; IF (new.purchase_date > CURRENT_DATE OR new.mfg_date >  
CURRENT_DATE) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = err_msg1; ELSEIF new.mfg_date > new.purchase_date THEN SIGNAL SQLSTATE  
'45000' SET MESSAGE_TEXT = err_msg2; ELSEIF new.pqty < 50 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = err_msg3; END IF; END;

```

Below the code editor are buttons for [Edit inline], [Edit], and [Create PHP code].

This screenshot shows the MySQL Workbench interface with the SQL tab selected. The query input field contains the following SQL statement:

```
1 | INSERT INTO stock VALUES (86934, '2022-11-01', '2022-12-01', 50);
```

Below the input field are buttons for Clear, Format, and Get auto-saved query. There is also a checkbox for Bind parameters.

The bottom section displays an error message in a pink box:

Error

SQL query: [Copy](#)

INSERT INTO stock VALUES (86934, '2022-11-01', '2022-12-01', 50);

MySQL said: #1644 - Mfg/Purchase Date must be before the current date

Server: MySQL:3306 » Database: inventory_management_system

Structure SQL Search Query Export Import

Run SQL query/queries on database inventory_management_system.

```
1 INSERT INTO stock VALUES (86934, '2022-12-11', '2022-12-01', 50);
```

Clear Format Get auto-saved query

Bind parameters

Delimiter ; Show this query here again Retain query box Rollback w

Error

SQL query: [Copy](#)

```
INSERT INTO stock VALUES (86934, '2022-12-11', '2022-12-01', 50);
```

MySQL said: [?](#)

```
#1644 - Mfg/Purchase Date must be before the current date
```

Server: MySQL:3306 » Database: inventory_management_system

Structure SQL Search Query Export Import

Run SQL query/queries on database inventory_management_system.

```
1 INSERT INTO stock VALUES (86934, '2022-10-01', '2022-11-01', 50);
```

Clear Format Get auto-saved query

Bind parameters

Delimiter ; Show this query here again Retain query box Rollback w

Error

SQL query: [Copy](#)

```
INSERT INTO stock VALUES (86934, '2022-10-01', '2022-11-01', 50);
```

MySQL said: [?](#)

```
#1644 - Mfg Date should be before Purchase Date
```

The screenshot shows the MySQL Workbench interface. The top bar indicates the server is MySQL: MySQL:3306 and the database is inventory_management_system. Below the bar are tabs for Structure, SQL, Search, Query, Export, Import, and a Help icon. The SQL tab is selected. A query window titled "Run SQL query/queries on database inventory_management_system:" contains the following SQL code:

```
1 INSERT INTO stock VALUES (86934, '2022-11-10', '2022-11-01', 5);
```

Below the query window are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Delimiter (set to ;), Show this query here again, Retain query box, and Rollback when finished.

A pink error box is displayed below the query window, titled "Error". It contains the following information:

SQL query: [Copy](#)

```
INSERT INTO stock VALUES (86934, '2022-11-10', '2022-11-01', 5);
```

MySQL said: [Copy](#)

Console product quantity should be minimum 50

b. Procedure with a Cursor to delete entries in vendor table with NULL values for product and customer that have at least one entry in the invoice table.

Procedure creation:

```
DELIMITER $  
CREATE PROCEDURE del_null()  
BEGIN  
    DECLARE vid int;  
    DECLARE finished INTEGER DEFAULT 0;  
    DECLARE del_null CURSOR FOR  
        SELECT DISTINCT vendor.vendor_id FROM vendor INNER JOIN  
        invoice WHERE vendor.vendor_id = invoice.vendor_id and  
        vendor.product_id IS NULL;  
  
    DECLARE CONTINUE HANDLER  
    FOR NOT FOUND SET finished = 1;
```

```

OPEN del_null;

    getRows: LOOP
        FETCH del_null INTO vid;
        IF finished THEN
            LEAVE getRows;
        END IF;
        DELETE FROM vendor WHERE vendor.vendor_id = vid AND
        vendor.product_id IS NULL;
    END LOOP;
    CLOSE del_null;
END $

DELIMITER ;

```

Procedure Call:

```
CALL del_null();
```

Screenshots:

This screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A message box at the top indicates that MySQL returned an empty result set. The SQL code area contains the definition of the 'del_null()' procedure, which uses a cursor to find vendor IDs with null product IDs and then deletes them. There are buttons for 'Edit inline', 'Edit', and 'Create PHP code' below the code.

```

CREATE PROCEDURE del_null() BEGIN DECLARE vid int; DECLARE finished INTEGER DEFAULT 0; DECLARE del_null CURSOR FOR SELECT DISTINCT
vendor.vendor_id FROM vendor INNER JOIN invoice WHERE vendor.vendor_id = invoice.vendor_id and vendor.product_id IS NULL; DECLARE CONTINUE
HANDLER FOR NOT FOUND SET finished = 1; OPEN del_null; getRows: LOOP FETCH del_null INTO vid; IF finished THEN LEAVE getRows; END IF;
DELETE FROM vendor WHERE vendor.vendor_id = vid AND vendor.product_id IS NULL; END LOOP; CLOSE del_null; END;

```

This screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A message box at the top indicates that MySQL returned an empty result set. The SQL code area contains the 'CALL del_null();' command. There are buttons for 'Edit inline', 'Edit', and 'Create PHP code' below the code.

```

CALL del_null();

```

Before procedure call:

Server: MySQL:3306 » Database: inventory_management_system » Table: vendor						
Browse		Structure		SQL		Search
fname	Iname	vendor_id	email_id	product_id	cust_id	
harry	singh	75071	harry@gmail.com	59854	26864	
harry	singh	75071	harry@gmail.com	59854	58462	
harry	singh	75071	harry@gmail.com	53028	57317	
harry	singh	75071	harry@gmail.com	14270	58462	
harry	singh	75071	harry@gmail.com	59854	63698	
harry	singh	75071	harry@gmail.com	98390	63698	
ramesh	sharma	40101	ramesh@gmail.com	59854	33295	
ramesh	sharma	40101	ramesh@gmail.com	15292	33295	
ramesh	sharma	40101	ramesh@gmail.com	59854	33295	
ramesh	sharma	40101	ramesh@gmail.com	14270	22150	
ramesh	sharma	40101	ramesh@gmail.com	59854	22150	
ram	c	70978	ram@gmail.com	53028	76443	
ram	c	70978	ram@gmail.com	59854	76443	
ranjeeta	s	94138	ranjeeta@gmail.com	59854	45406	
ranjeeta	s	94138	ranjeeta@gmail.com	45124	26864	
ranjeeta	s	94138	ranjeeta@gmail.com	98390	45406	
rakesh	sharma	29690	rakesh@gmail.com	81010	43143	
rakesh	sharma	29690	rakesh@gmail.com	15292	43143	
vishwa	mehta	73660	vis@gmail.com	NULL	NULL	
vishwa	mehta	73660	vis@gmail.com	59854	58462	
vismaya	r	84992	vismaya@gmail.com	NULL	NULL	

After procedure call:

Server: MySQL:3306 » Database: inventory_management_system » Table: vendor						
Browse		Structure		SQL		Search
fname	Iname	vendor_id	email_id	product_id	cust_id	
narry	singh	75071	narry@gmail.com	59854	57317	
harry	singh	75071	harry@gmail.com	59854	26864	
harry	singh	75071	harry@gmail.com	59854	58462	
harry	singh	75071	harry@gmail.com	53028	57317	
harry	singh	75071	harry@gmail.com	14270	58462	
harry	singh	75071	harry@gmail.com	59854	63698	
harry	singh	75071	harry@gmail.com	98390	63698	
ramesh	sharma	40101	ramesh@gmail.com	59854	33295	
ramesh	sharma	40101	ramesh@gmail.com	15292	33295	
ramesh	sharma	40101	ramesh@gmail.com	59854	33295	
ramesh	sharma	40101	ramesh@gmail.com	14270	22150	
ramesh	sharma	40101	ramesh@gmail.com	59854	22150	
ram	c	70978	ram@gmail.com	53028	76443	
ram	c	70978	ram@gmail.com	59854	76443	
ranjeeta	s	94138	ranjeeta@gmail.com	59854	45406	
ranjeeta	s	94138	ranjeeta@gmail.com	45124	26864	
ranjeeta	s	94138	ranjeeta@gmail.com	98390	45406	
rakesh	sharma	29690	rakesh@gmail.com	81010	43143	
rakesh	sharma	29690	rakesh@gmail.com	15292	43143	
vishwa	mehta	73660	vis@gmail.com	59854	58462	
vismaya	r	84992	vismaya@gmail.com	NULL	NULL	

11. Developing a Frontend

1. app.py

```
import streamlit as st
from database import check_login, add_login

def main():
    st.title("Inventory Management System")
    col1, col2 = st.columns(2)
    with col1:
        st.subheader("Login")
        email0 = st.text_input("Email: ")
        password0 = st.text_input("Password: ", type = "password")
        if st.button("Login"):
            check_login(email0, password0)
    with col2:
        st.subheader("SignUp:")
        fname = st.text_input("First Name:")
        lname = st.text_input("Last Name")
        email = st.text_input("Email:")
        password = st.text_input("Password:", type = "password")
        if st.button("Signup"):
            add_login(fname, lname, email, password)

if __name__ == '__main__':
    main()
```

2. database.py

```
import streamlit as st
import mysql.connector
import random
```

```

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    password = "",
    database = "inventory_management_system"
)
c = mydb.cursor()

from reload import reload
def check_login(email, password):
    # print(email)
    values = email
    c.execute("SELECT * FROM login WHERE email_id = %s", (values,))
    passws = c.fetchall()
    for passw in passws:
        if passw[1] == password:
            reload("home")
            break
    else:
        st.error("Incorrect credentials, please try again.")

def add_login(fn, ln, email, password):
    values = email
    c.execute("SELECT * FROM login WHERE email_id = %s", (values,))
    x = c.fetchall()
    if x != []:
        print(x)
        st.error("User already exists :/")
    else:

```

```

        c.execute("INSERT INTO login (email_id, pass) VALUES
(%s,%s)", (email, password))

        vid = random.randint(10000, 99999)

        c.execute("INSERT INTO vendor (fname, lname, vendor_id,
email_id) VALUES (%s, %s, %s, %s)", (fn, ln, vid, email))

        st.success("Successfully Added User")

# else:

#     st.error("User already exists :/")

def update_password(mail, old_pass, new_pass):

    c.execute("UPDATE login SET pass = %s WHERE email_id = %s AND
pass = %s", (new_pass, mail, old_pass))

    if c.rowcount is not None:

        st.success("Password updated successfully!")

def view_stock():

    c.execute("SELECT * FROM stock")

    data = c.fetchall()

    return data

def add_stock(pid, pdate, mdate, qty):

    c.execute("INSERT INTO stock VALUES (%s, %s, %s, %s)", (pid,
pdate, mdate, qty))

    st.success("Values inserted")

def prod_not_in_stock():

    c.execute("SELECT product.product_id FROM product WHERE
product.product_id NOT IN (SELECT stock.pid FROM stock)")

    data = c.fetchall()

    return data

def prod_in_stock():

```

```

        c.execute("SELECT product.product_id FROM product WHERE
product.product_id IN (SELECT stock.pid FROM stock)")

        data = c.fetchall()

        return data


def get_product(pid):
    c.execute("SELECT * FROM product WHERE product.product_id =
{}".format(pid))

    data = c.fetchall()

    print(data)

    return data


def get_stock(pid):
    c.execute("SELECT * FROM stock WHERE stock.pid =
{}".format(pid))

    data = c.fetchall()

    print(data)

    return data


def update_stock(pid, pdate, mdate, qty):
    c.execute("UPDATE stock SET purchase_date = %s, mfg_date = %s,
qty = %s WHERE pid = %s", (pdate, mdate, qty, pid))

    st.success("Stock updated successfully")


def remove_stock(pid):
    c.execute("DELETE FROM stock WHERE stock.pid = {}".format(pid))

    st.success("Product Deleted successfully")


def view_product():
    c.execute("SELECT * FROM product")

    data = c.fetchall()

    return data

```

```

def add_product(pid, pname, cp, mfg, mrp):
    c.execute("INSERT INTO product VALUES (%s, %s, %s, %s, %s)", (pid, pname, cp, mfg, mrp))
    st.success("Inserted Product")

def get_product_names():
    c.execute("SELECT product.product_name FROM product")
    data = c.fetchall()
    return data

def get_product_id(pname):
    value = pname
    c.execute("SELECT product.product_id FROM product WHERE product_name = %s", (value,))
    data = c.fetchall()
    return data[0][0]

def get_prod_mrp(pid):
    value = pid
    c.execute("SELECT product.mrp FROM product WHERE product.product_id = %s", (value,))
    data = c.fetchall()
    return data[0][0]

def update_prod_price(pid, cp):
    c.execute("UPDATE product SET product.cost_price = %s WHERE product.product_id = %s", (cp, pid))
    st.success("Cost price updated")

def view_customer():
    c.execute("SELECT * FROM customer")

```

```

        data = c.fetchall()
        return data

def add_customer(cid, pn, mail, cname, sno, sname, pin):
    c.execute("INSERT INTO customer VALUES (%s, %s, %s, %s, %s, %s, %s)", (cid, pn, mail, cname, sno, sname, pin))
    st.success("Inserted Customer")

def get_customer(cid):
    c.execute("SELECT * FROM customer WHERE customer.cust_id = {}".format(cid))
    data = c.fetchall()
    print(data)
    return data

def get_cust_names():
    c.execute("SELECT customer.cust_name FROM customer")
    data = c.fetchall()
    return data

def get_cust_ids(cname):
    value = cname
    c.execute("SELECT customer.cust_id FROM customer WHERE customer.cust_name = %s", (value,))
    data = c.fetchall()
    return data

def update_customer(cid, phno, email, cname, stno, stname, pin):
    c.execute("UPDATE customer SET phone_no = %s, cust_email_id = %s, cust_name = %s, str_no = %s, str_name = %s, pincode = %s WHERE cust_id = %s", (phno, email, cname, stno, stname, pin, cid))
    st.success("Successfully Updated Customer")

```

```

def view_invoice():
    c.execute("SELECT * FROM invoice")
    data = c.fetchall()
    return data

def get_vendor_id(mail):
    print(mail)
    c.execute("SELECT DISTINCT vendor.vendor_id FROM vendor WHERE
    vendor.email_id = %s", (mail, ))
    data = c.fetchall()
    print("data", data[0][0])
    return data[0][0]

def add_invoice(pid, ino, idate, sp, pqty, dis, phno, vid):
    c.execute("INSERT INTO invoice VALUES (%s, %s, %s, %s, %s, %s,
    %s, %s)", (pid, ino, idate, sp, pqty, dis, phno, vid))
    c.execute("SELECT customer.cust_id FROM customer WHERE
    customer.phone_no = %s", (phno,))
    cid = c.fetchall()[0][0]
    c.execute("SELECT DISTINCT vendor.fname, vendor.lname,
    vendor.vendor_id, vendor.email_id FROM vendor WHERE
    vendor.vendor_id = %s", (vid,))
    d = c.fetchall()
    fn, ln, vid0, mail = d[0][0], d[0][1], d[0][2], d[0][3]
    c.execute("INSERT INTO vendor VALUES (%s, %s, %s, %s, %s, %s)",
    (fn, ln, vid0, mail, pid, cid))
    st.success("Invoice inserted successfully")

def get_invoice(iid):
    c.execute("SELECT * FROM invoice WHERE invoice.invoice_no =
    {}".format(iid))
    data = c.fetchall()
    return data

```

```
def get_phnos():
    c.execute("SELECT customer.phone_no FROM customer")
    data = c.fetchall()
    return data
```

3. home.py

```
import streamlit as st
from database import *
import pandas as pd
import random

st.title("Welcome to ListAll")

def main():
    menu = ["View Stock", "Add item to stock", "Update Stock",
    "Remove Stock", "View Products", "Add Product", "Update Product
    Price", "View Customers", "Add Customer", "Update Customer", "View
    Invoice", "Add Invoice", "Change Password"]

    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "View Stock":
        st.subheader("Current Stock:")
        data = view_stock()
        df = pd.DataFrame(data, columns = ['pid', 'purchase_date',
        'mfg_date', 'pqty'])
        st.dataframe(df)

    elif choice == "Add item to stock":
        st.subheader("Add New Product to Current Stock:")
        list_of_products = [i[0] for i in prod_not_in_stock()]
        prod = st.selectbox("Products that can be added to stock",
        list_of_products)
        result = get_product(prod)
```

```

if result:
    pid = st.text_input("Product ID:", result[0][0])
    pname = st.text_input("Product Name:", result[0][1])
    pur_date = st.text_input("Purchase Date(YYYY-MM-DD):")
    mfg_date = st.text_input("Manufacturing Date(YYYY-MM-"
DD):")

    qty = st.text_input("Product Quantity:")
    if st.button("Add"):
        add_stock(pid, pur_date, mfg_date, qty)
with st.expander("View Updated table"):
    data = view_stock()
    df = pd.DataFrame(data, columns = ['pid',
'purchase_date', 'mfg_date', 'pqty'])
    st.dataframe(df)

elif choice == "Update Stock":
    st.subheader("Update Stock:")
    list_of_products = [i[0] for i in prod_in_stock()]
    prod = st.selectbox("Products that can be updated in
stock", list_of_products)
    result = get_stock(prod)
    res = get_product(prod)
    # print(result[0])
    if result:
        # print(type(result[0][1]), type(result[0][2]))
        pid = result[0][0]
        pdate = result[0][1]
        mdate = result[0][2]
        pqty = result[0][3]
        col1, col2= st.columns(2)
        with col1:

```

```

        pid0 = st.text_input("Product ID:", pid, disabled =
True)

        new_pdate = st.text_input("Purchase Date(YYYY-MM-
DD):", pdate)

        with col2:

            pname = st.text_input("Product Name:", res[0][1],
disabled = True)

            new_mdate = st.text_input("Manufacturing Date(YYYY-
MM-DD):", mdate)

            new_pqty = st.text_input("Product Quantity:", pqty)

            if st.button("Update"):

                update_stock(pid0, new_pdate, new_mdate, new_pqty)

            with st.expander("View Updated table"):

                data = view_stock()

                df = pd.DataFrame(data, columns = ['pid',
'purchase_date', 'mfg_date', 'pqty'])

                st.dataframe(df)

    elif choice == "Remove Stock":

        st.subheader("Remove Stock:")

        list_of_products = [i[0] for i in prod_in_stock()]

        prod = st.selectbox("Products that can be added to stock",
list_of_products)

        result = get_product(prod)

        if result:

            pid = result[0][0]

            if st.button("Remove "+str(result[0][1])):

                remove_stock(pid)

            with st.expander("View Updated table"):

                data = view_stock()

                df = pd.DataFrame(data, columns = ['pid',
'purchase_date', 'mfg_date', 'pqty'])

                st.dataframe(df)

```

```

        elif choice == "View Products":
            st.subheader("Products:")
            data = view_product()
            df = pd.DataFrame(data, columns = ['product_id',
            'product_name', 'cost_price', 'manufacturer', 'mrp'])
            st.dataframe(df)

        elif choice == "Add Product":
            st.subheader("Enter product details:")
            gen_pid = random.randint(10000, 99999)
            d = get_product(gen_pid)
            if not d:
                print(gen_pid)
                pid = st.text_input("Product ID:", gen_pid, disabled =
True)
            else:
                st.error("Please reload the page.")
                pname = st.text_input("Product Name:")
                cp = st.text_input("Cost Price:")
                mfg = st.text_input("Manufacturer Name:")
                mrp = st.text_input("MRP:")
                if st.button("Add"):
                    add_product(pid, pname, cp, mfg, mrp)
            with st.expander("View Updated table"):
                data = view_product()
                df = pd.DataFrame(data, columns = ['product_id',
                'product_name', 'cost_price', 'manufacturer', 'mrp'])
                st.dataframe(df)

        elif choice == "Update Product Price":
            st.subheader("Update Product:")

```

```

list_of_products = [i[0] for i in get_product_names()]
pname = st.selectbox("Product Name:", list_of_products)
prod = get_product_id(pname)
result = get_product(prod)
# print(result[0])
if result:
    # print(type(result[0][1]), type(result[0][2]))
    pid = result[0][0]
    pname = result[0][1]
    cp = result[0][2]
    mfg = result[0][3]
    mrp = result[0][4]
    col1, col2= st.columns(2)
    with col1:
        pid0 = st.text_input("Product ID:", pid, disabled =
True)
        pname0 = st.text_input("Name:", pname, disabled =
True)
        mrp0 = st.text_input("MRP:", mrp, disabled = True)
    with col2:
        new_cp = st.text_input("Cost Price:", cp)
        mfg0 = st.text_input("Manufacturer Name:", mfg,
disabled = True)
    if st.button("Update"):
        update_prod_price(pid0, new_cp)
    with st.expander("View Updated table"):
        data = view_product()
        df = pd.DataFrame(data, columns = ['product_id',
'product_name', 'cost_price', 'manufacturer', 'mrp'])
        st.dataframe(df)

elif choice == "View Customers":

```

```

st.subheader("Customers:")
data = view_customer()
df = pd.DataFrame(data, columns = ['cust_id', 'phone_no',
'cust_email_id', 'cust_name', 'str_no', 'str_name', 'pincode'])
st.dataframe(df)

elif choice == "Add Customer":
    st.subheader("Enter customer details:")
    gen_cid = random.randint(10000, 99999)
    d = get_customer(gen_cid)
    if not d:
        print(gen_cid)
        cid = st.text_input("Customer ID:", gen_cid, disabled =
True)
    else:
        st.error("Please reload the page.")
    cname = st.text_input("Customer Name:")
    pn = st.text_input("Phone Number:")
    mail = st.text_input("Email ID:")
    col1, col2, col3 = st.columns(3)
    with col1:
        sno = st.text_input("Street Number:")
    with col2:
        sname = st.text_input("Street Name:")
    with col3:
        pin = st.text_input("Pincode:")

if st.button("Add"):
    add_customer(cid, pn, mail, cname, sno, sname, pin)

with st.expander("View Updated table"):

```

```

        data = view_customer()

        df = pd.DataFrame(data, columns = ['cust_id',
'phone_no', 'cust_email_id', 'cust_name', 'str_no', 'str_name',
'pincode'])

        st.dataframe(df)

    elif choice == "Update Customer":

        st.subheader("Update Customer:")

        list_of_cnames = [i[0] for i in get_cust_names()]

        # print(list_of_cnames)

        cname = st.selectbox("Select customer to be updated",
list_of_cnames)

        # print(cname)

        cid = get_cust_ids(cname)

        # print(cid[0][0])

        result = get_customer(cid[0][0])

        # print(result[0])

        if result:

            cid = result[0][0]

            phno = result[0][1]

            email = result[0][2]

            cname = result[0][3]

            stno = result[0][4]

            stname = result[0][5]

            pin = result[0][6]

            cid0 = st.text_input("Customer ID:", cid, disabled =
True)

            new_phno = st.text_input("Phone Number", phno)

            new_email = st.text_input("Email ID:", email)

            new_cname = st.text_input("Name:", cname)

            new_stno = st.text_input("Street Number:", stno)

            new_stname = st.text_input("Street Name:", stname)

```

```

        new_pin = st.text_input("Pincode:", pin)
        if st.button("Update"):
            update_customer(cid0, new_phno, new_email,
new_cname, new_stno, new_stname, new_pin)

        with st.expander("View Updated table"):
            data = view_customer()
            df = pd.DataFrame(data, columns = ['cust_id',
'phone_no', 'cust_email_id', 'cust_name', 'str_no', 'str_name',
'pincode'])
            st.dataframe(df)

    elif choice == "View Invoice":
        st.subheader("Invoice:")
        data = view_invoice()
        df = pd.DataFrame(data, columns = ['prod_id', 'invoice_no',
'invoice_date', 'selling_price', 'prod_qty', 'discount',
'phone_no', 'vendor_id'])
        st.dataframe(df)

    elif choice == "Add Invoice":
        st.subheader("Add Invoice:")
        list_of_products = [i[1] for i in view_product()]
        prod = st.selectbox("Product Name:", list_of_products)
        pid = get_product_id(prod)
        gen_iid = random.randint(10000, 99999)
        d = get_invoice(gen_iid)
        if not d:
            print(gen_iid)
            iid = st.text_input("Invoice ID:", gen_iid, disabled =
True)
        else:
            st.error("Please reload the page.")
            idate = st.text_input("Invoice Date (YYYY-MM-DD):")

```

```

        dis = st.text_input("Discount Given(%):")
        mrp = get_prod_mrp(pid)
        if dis:
            dis = float(dis) * 0.01
            sp = mrp - (mrp * dis)
            sp0 = st.text_input("Selling Price:", sp, disabled =
True)

        pqty = st.text_input("Product Quantity:")
        list_of_phnos = [i[0] for i in get_phnos()]
        pno = st.selectbox("Customer Phone Number:", list_of_phnos)
        mail = st.text_input("Vendor Email ID:")
        # vid = 0
        if mail:
            vid = get_vendor_id(mail)
            # pass
            # phno = get_cus_phone(pno)
        if st.button("Add Invoice"):
            add_invoice(pid, iid, idate, sp0, pqty, dis, pno, vid)
        with st.expander("View Updated Invoice"):
            data = view_invoice()
            df = pd.DataFrame(data, columns = ['prod_id',
'invoice_no', 'invoice_date', 'selling_price', 'prod_qty',
'discount', 'phone_no', 'vendor_id'])
            st.dataframe(df)

    elif choice == "Change Password":
        st.subheader("Enter New Credentials Below:")
        email = st.text_input("Email:")
        old_pass = st.text_input("Old Password:", type =
"password")
        new_pass = st.text_input("New Password:", type =
"password")
        if st.button("Change Password"):

```

```
        update_password(email, old_pass, new_pass)

    else:
        st.subheader("Home Page")
if __name__ == '__main__':
    main()
```

4. reload.py

```
import streamlit as st
import os
import keyboard
import threading
import time
wait_second = 10
def threadFunc():
    time.sleep(wait_second)

def reload(page):
    th = threading.Thread(target=threadFunc)
    th.start()
    os.system(r"streamlit run " + page + ".py")
    th.join()
```

Screenshots:

The screenshot shows a Streamlit application running at localhost:8501. The title bar indicates "localhost / MySQL / inventory_m" and "app - Streamlit". The main content area has a dark background with white text. It features two forms side-by-side: a "Login" form and a "SignUp:" form. The "Login" form includes fields for Email (with value "vis@gmail.com") and Password (with placeholder "*****"). A red-bordered "Login" button is positioned below these fields. The "SignUp:" form includes fields for First Name, Last Name, Email, and Password, followed by a "Signup" button.

The screenshot shows the Streamlit application at localhost:8501. The title bar indicates "localhost / MySQL / inventory_m" and "home - Streamlit". On the left, there is a sidebar with a "Menu" section containing a dropdown menu with "View Stock" selected. The main content area has a dark background with white text. The title "Welcome to ListAll" is displayed prominently. Below it, the heading "Current Stock:" is shown. A table titled "Current Stock:" displays the following data:

	pid	purchase_date	mfg_date	pqty
0	59854	2022-10-21	2022-10-21	80
1	53028	2022-10-18	2022-10-18	90
2	56145	2022-10-12	2022-10-10	65
3	81010	2022-10-21	2022-10-21	60
4	99016	2022-10-17	2022-10-16	50
5	15292	2022-11-17	2022-10-19	50
6	45124	2022-11-17	2022-09-29	50
7	14270	2022-11-17	2022-09-20	100

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Welcome to ListAll

Add New Product to Current Stock:

Products that can be added to stock

80811

Product ID:

80811

Product Name:

Bread

Purchase Date(YYYY-MM-DD):

2022-11-05

Manufacturing Date(YYYY-MM-DD):

2022-11-03

Product Quantity:

50

Add

Values inserted

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

50

Add

Values inserted

View Updated table

	pid	purchase_date	mfg_date	qty
0	59854	2022-10-21	2022-10-21	80
1	53028	2022-10-18	2022-10-18	90
2	56145	2022-10-12	2022-10-10	65
3	81010	2022-10-21	2022-10-21	60
4	99016	2022-10-17	2022-10-16	50
5	15292	2022-11-17	2022-10-19	50
6	45124	2022-11-17	2022-09-29	50
7	14270	2022-11-17	2022-09-20	100
8	80811	2022-11-05	2022-11-03	50

Made with Streamlit

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Menu

Update Stock

Welcome to ListAll

Update Stock:

Products that can be updated in stock

80811

Product ID: 80811 Product Name: Bread

Purchase Date(YYYY-MM-DD): 2022-11-05 Manufacturing Date(YYYY-MM-DD): 2022-11-03

Product Quantity: 90

Update

Stock updated successfully

View Updated table

Stock updated successfully

View Updated table

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Menu

Update Stock

90

Update

Stock updated successfully

View Updated table

	pid	purchase_date	mfld_date	qty
0	59854	2022-10-21	2022-10-21	80
1	53028	2022-10-18	2022-10-18	90
2	56145	2022-10-12	2022-10-10	65
3	81010	2022-10-21	2022-10-21	60
4	99016	2022-10-17	2022-10-16	50
5	15292	2022-11-17	2022-10-19	50
6	45124	2022-11-17	2022-09-29	50
7	14270	2022-11-17	2022-09-20	100
8	80811	2022-11-05	2022-11-03	90

Made with Streamlit

	pid	purchase_date	mfld_date	qty
0	59854	2022-10-21	2022-10-21	80
1	53028	2022-10-18	2022-10-18	90
2	56145	2022-10-12	2022-10-10	65
3	81010	2022-10-21	2022-10-21	60
4	99016	2022-10-17	2022-10-16	50
5	15292	2022-11-17	2022-10-19	50
6	45124	2022-11-17	2022-09-29	50
7	14270	2022-11-17	2022-09-20	100
8	80811	2022-11-05	2022-11-03	90

localhost / MySQL / inventory_m x | app - Streamlit x home - Streamlit x +

Gmail PES University Email addresses us...

Welcome to ListAll

Remove Stock:

Products that can be added to stock

80811

Remove Bread

Product Deleted successfully

View Updated table

	pid	purchase_date	mfng_date	qty
0	59854	2022-10-21	2022-10-21	80
1	53028	2022-10-18	2022-10-18	90
2	56145	2022-10-12	2022-10-10	65
3	81010	2022-10-21	2022-10-21	60
4	99016	2022-10-17	2022-10-16	50
5	15292	2022-11-17	2022-10-19	50
6	45124	2022-11-17	2022-09-29	50
7	14270	2022-11-17	2022-09-20	100

localhost / MySQL / inventory_m x | app - Streamlit x home - Streamlit x +

Gmail PES University Email addresses us...

Welcome to ListAll

Products:

	product_id	product_name	cost_price	manufacturer	mrp
1	53028	curd	30.0000	Amul	34.0000
2	56145	butter	25.0000	Amul	28.0000
3	81010	cow milk	15.0000	Nandini	17.0000
4	99016	yogurt	25.0000	Nandini	30.0000
5	15292	paneer	55.0000	Milky Mist	60.0000
6	45124	cheese	85.0000	Amul	90.0000
7	14270	ghee	440.0000	Nandini	450.0000
8	86934	ice cream family	180.0000	Quality Walls	200.0000
9	98390	Ice cream cone	35.0000	Cornetto	40.0000
10	80811	Bread	45.0000	Britania	55.0000

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Welcome to ListAll

Enter product details:

Product ID: 83067

Product Name: curd

Cost Price: 32

Manufacturer Name: Milky Mist

MRP: 37

Add

Inserted Product

View Updated table

Product ID: 83067

Product Name: curd

Cost Price: 32

Manufacturer Name: Milky Mist

MRP: 37

Add

Inserted Product

View Updated table

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Add

Inserted Product

View Updated table

	product_id	product_name	cost_price	manufacturer	mrp
2	56145	butter	25.0000	Amul	28.0000
3	81010	cow milk	15.0000	Nandini	17.0000
4	99016	yogurt	25.0000	Nandini	30.0000
5	15392	paneer	55.0000	Milky Mist	60.0000
6	45124	cheese	85.0000	Amul	90.0000
7	14270	ghee	440.0000	Nandini	450.0000
8	86934	ice cream family	180.0000	Quality Walls	200.0000
9	98390	ice cream cone	35.0000	Cornetto	40.0000
10	80811	Bread	45.0000	Britannia	55.0000
11	83067	curd	32.0000	Milky Mist	37.0000

Made with Streamlit

product_id: 83067

product_name: curd

cost_price: 32

manufacturer: Milky Mist

mrp: 37

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Welcome to ListAll

Update Product:

Product Name: curd

Product ID: 53028 Cost Price: 31.0

Name: curd Manufacturer Name: Amul

MRP: 34.0

Update

Cost price updated

[View Updated table](#)

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Update

Cost price updated

[View Updated table](#)

	product_id	product_name	cost_price	manufacturer	mrp
0	59854	milk	17.0000	Amul	19.0000
1	53028	curd	31.0000	Amul	34.0000
2	56145	butter	25.0000	Amul	28.0000
3	81010	cow milk	15.0000	Nandini	17.0000
4	99016	yogurt	25.0000	Nandini	30.0000
5	15292	paneer	55.0000	Milky Mist	60.0000
6	45124	cheese	85.0000	Amul	90.0000
7	14270	ghee	440.0000	Nandini	450.0000
8	86934	ice cream family	180.0000	Quality Walls	200.0000
9	98390	ice cream cone	35.0000	Cornetto	40.0000

Made with Streamlit

localhost / MySQL / inventory_m | app - Streamlit | home - Streamlit | +

Gmail PES University Email addresses us... Other bookmarks

Welcome to ListAll

Customers:

	cust_id	phone_no	cust_email_id	cust_name	str_no	str_name	pincode
0	57317	528998172	inderpal@gmail.com	Inderpal Ravinder	12	Goel Street	64972
1	26864	804384922	ishi@gmail.com	Ishita Devraj	14	Ramkisson Street	42742
2	58462	8545588678	anilip@gmail.com	Anik Prabhakar	56	Sekhon Street	13984
3	63698	3816889497	ninadran@gmail.com	Ninad Ram	37	Kart Street	45663
4	33295	9489033854	devis@yahoo.com	Devi Sunder	28	Subramaniam Street	37692
5	22150	9056364822	rama123@yahoo.com	Ramadevi Gopala	57	Sen Street	75808
6	76443	7294778333	pravinajaya@gmail.com	Pravina Jaya	22	Raja Street	59164
7	45408	4305436406	anilabhi@gmail.com	Anil Abhishek	71	Shroff Street	23474
8	26865	8145956923	nilofara@hotmail.com	Nilofar Amardeep	83	Vig Street	15865
9	43143	5636459498	vinayarvind24@hotmail.c	Vinay Arvind	26	Chakraborty Street	52534
10	13261	9242192421	abc123@gmail.com	Niyanta	204	Shobha Classic	560006

localhost / MySQL / inventory_m | app - Streamlit | home - Streamlit | +

Gmail PES University Email addresses us... Other bookmarks

Welcome to ListAll

Add Customer

Enter customer details:

Customer ID: 57652

Customer Name: Sunaina

Phone Number: 1234567890

Email ID: sunaina@gmail.com

Street Number: 12 Street Name: Shroff Street Pincode: 234740

Add

Inserted Customer

[View Updated table](#)

localhost / MySQL / inventory_m | app - Streamlit | home - Streamlit | +

Gmail PES University Email addresses us... Other bookmarks

Inserted Customer

Add

Menu

Add Customer

View Updated table

	cust_id	phone_no	cust_email_id	cust_name	str_no	str_name	pi
0	57317	5289998172	inderpal@gmail.com	Inderpal Ravinder	12	Goel Street	1
1	26864	8043894022	ish@gmail.com	Ishita Devraj	14	Ramkisson Street	1
2	58462	8545588678	anikp@gmail.com	Anik Prabhakar	56	Sekhon Street	1
3	63698	3816889497	ninadram@gmail.com	Ninad Ram	37	Kart Street	1
4	33295	9489133654	devi@yahoo.com	Devi Sunder	28	Subramanian Street	1
5	22150	9056364622	rama123@yahoo.com	Ramadevi Gopala	57	Sen Street	1
6	76443	7294778333	pravinjaya@gmail.com	Pravina Jaya	22	Raja Street	1
7	45406	4305436406	anilabhij@gmail.com	Anil Abhishek	71	Shroff Street	1
8	26865	8145956923	nilofera@hotmail.com	Nilofer Amandeep	83	Vig Street	1
9	43143	5636459498	vinayarvind24@hotmail.c	Vinay Aravind	26	Chakraborty Street	1
10	13261	9242192421	abc123@gmail.com	Niyanta	204	Shobha Classic	1
11	57652	1234567890	sunaina@gmail.com	Sunaina	12	Shroff Street	1

localhost / MySQL / inventory_m | app - Streamlit | home - Streamlit | +

Gmail PES University Email addresses us... Other bookmarks

Select customer to be updated

Customer ID: Sunaina

Customer ID: 57652

Phone Number: 1234567890

Email ID: sunaina@gmail.com

Name: Sunaina Mishra

Street Number: 12

Street Name: Shroff Street

Pincode: 234740

Update

Successfully Updated Customer

The screenshot shows a Streamlit application running at localhost:8502. The title bar indicates the app is titled "home - Streamlit". The main content area has a dark background. On the left, there's a sidebar with a "Menu" section containing a dropdown menu with "Update Customer" selected. The main area has a header "Update" with a green success message "Successfully Updated Customer". Below it is a table titled "View Updated table" showing 11 rows of customer data. The columns are: cust_id, phone_no, cust_email_id, cust_name, str_no, str_name, and pi. The data includes various names like Anik Prabhakar, Ninal Ram, Devi Sunder, etc., along with their respective addresses and phone numbers.

	cust_id	phone_no	cust_email_id	cust_name	str_no	str_name	pi
2	58462	854588678	anik@gmail.com	Anik Prabhakar	56	Sekhon Street	1
3	63698	381689497	ninalram@gmail.com	Ninal Ram	37	Kari Street	4
4	33295	948933654	devis@yahoo.com	Devi Sunder	28	Subramaniam Street	2
5	22150	9056364622	rama123@yahoo.com	Ramadevi Gopala	57	Sen Street	1
6	76443	7294778333	pravinajaya@gmail.com	Pravina Jaya	22	Raja Street	5
7	45406	4305436406	anilabhii@gmail.com	Anil Abhishek	73	Shroff Street	2
8	26865	8145956623	nilofara@hotmai.com	Nilofar Amardeep	83	Vig Street	1
9	43143	5636459498	vinayarvind24@hotmail.c	Vinay Aravind	26	Chakraborty Street	2
10	13261	9242192421	abc123@gmail.com	Niyanta	204	Shobha Classic	1
11	57652	1234567890	sunaina@gmail.com	Sunaina Mishra	12	Shroff Street	2

The screenshot shows a Streamlit application running at localhost:8502. The title bar indicates the app is titled "home - Streamlit". The main content area has a dark background. On the left, there's a sidebar with a "Menu" section containing a dropdown menu with "View Invoice" selected. The main area displays a welcome message "Welcome to ListAll" and a section titled "Invoice:" followed by a table showing invoice details. The columns are: prod_id, invoice_no, invoice_date, selling_price, prod_qty, discount, phone_no, and vendor_id. The data includes various invoices with dates ranging from 2022-10-22 to 2022-11-18, and quantities from 2 to 3.

	prod_id	invoice_no	invoice_date	selling_price	prod_qty	discount	phone_no	vendor_id
0	59854	38980	2022-10-22	15.2000	2	0.2000	5289998172	75071
1	59854	85957	2022-10-23	15.2000	2	0.2000	8043894922	75071
2	59854	99454	2022-10-23	15.2000	2	0.2000	8545588678	40101
3	59854	20453	2022-11-18	15.2000	3	0.2000	8545588678	73660
4	59854	49678	2022-11-18	15.2000	3	0.2000	8545588678	73660
5	59854	88070	2022-11-18	15.2000	3	0.2000	8545588678	73660
6	59854	69302	2022-11-18	15.2000	3	0.2000	8545588678	73660
7	59854	43769	2022-11-18	18.4300	3	0.0300	9242192421	74044

localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Product Name: Curd

Invoice ID: 85114

Invoice Date (YYYY-MM-DD): 2022-11-05

Discount Given(%): 4

Selling Price: 32.64

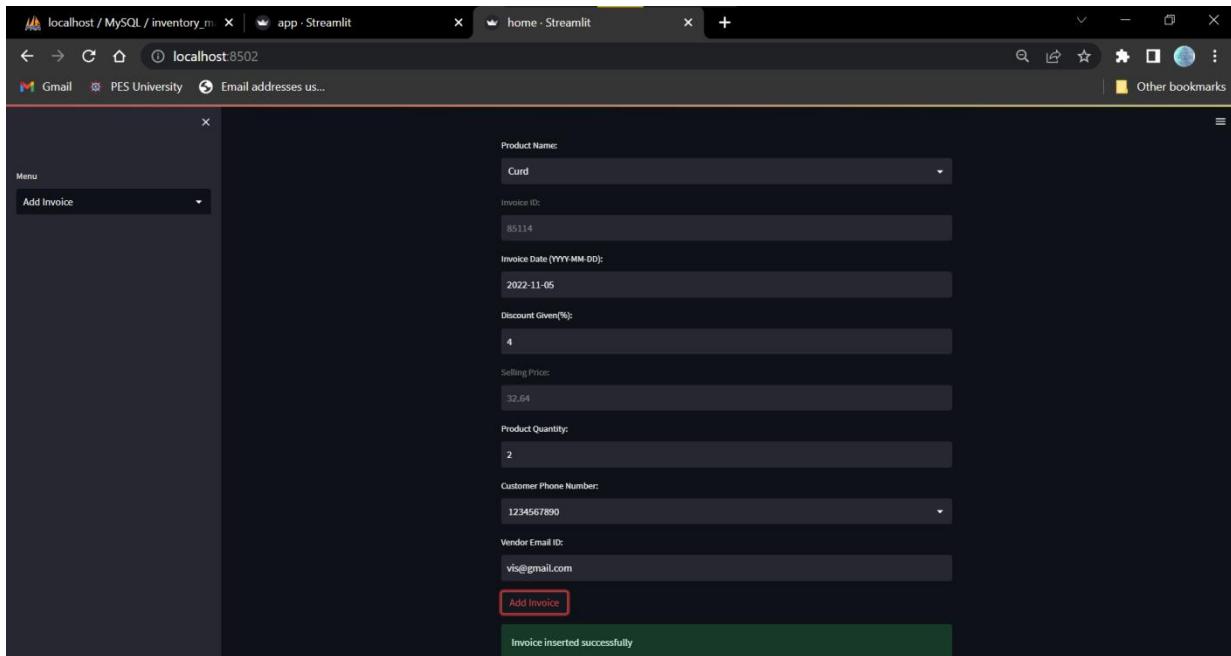
Product Quantity: 2

Customer Phone Number: 1234567890

Vendor Email Id: vis@gmail.com

Add Invoice

Invoice inserted successfully



localhost / MySQL / inventory_m app - Streamlit home - Streamlit +

Gmail PES University Email addresses us... Other bookmarks

Menu Add Invoice

vis@gmail.com

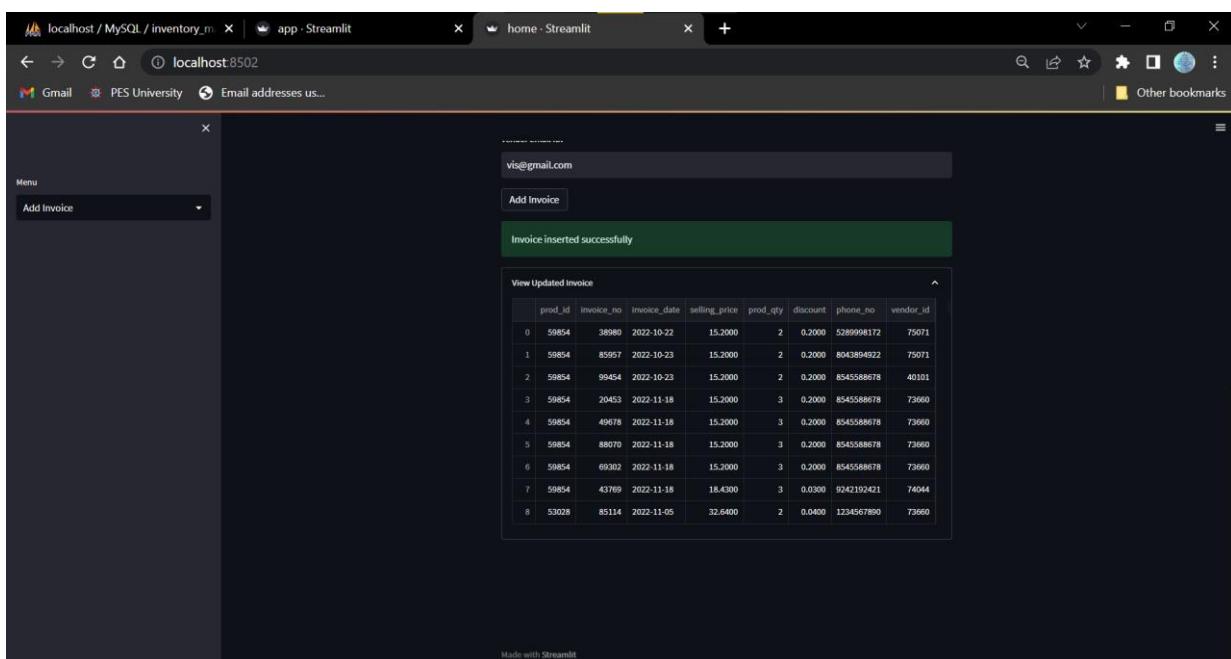
Add Invoice

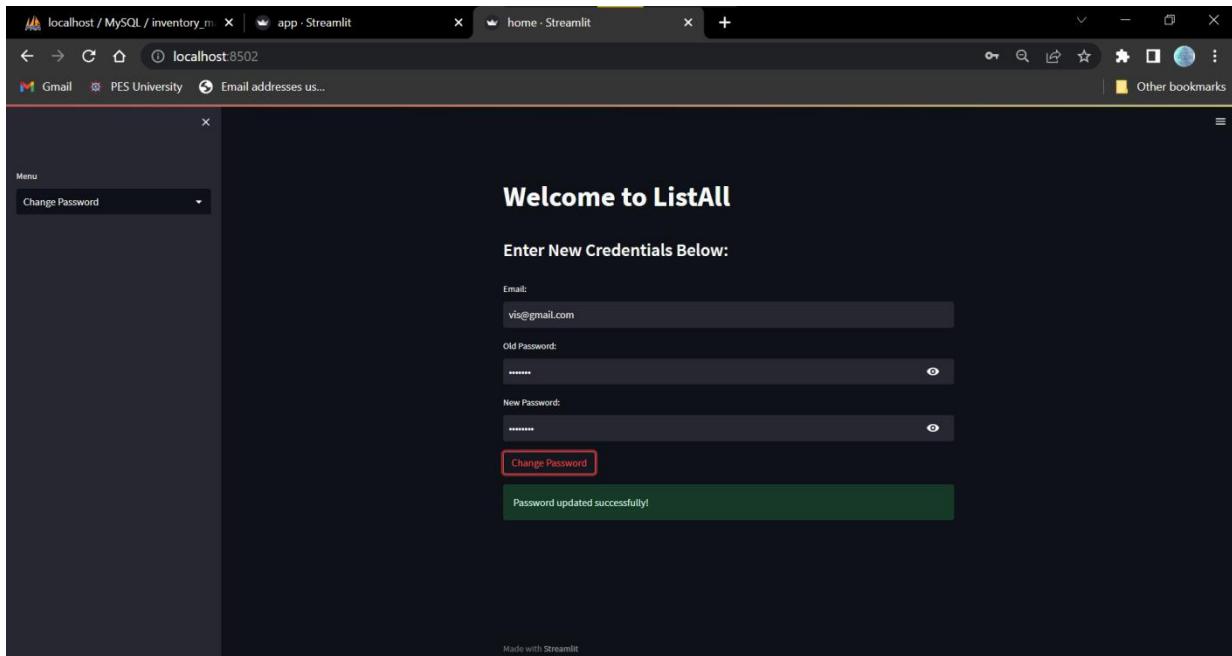
Invoice inserted successfully

View Updated Invoice

	prod_id	invoice_no	invoice_date	selling_price	prod_qty	discount	phone_no	vendor_id
0	59854	38980	2022-10-22	15.2000	2	0.2000	5289998172	75071
1	59854	85957	2022-10-23	15.2000	2	0.2000	8043894922	75071
2	59854	99454	2022-10-23	15.2000	2	0.2000	8545588678	40101
3	59854	20453	2022-11-18	15.2000	3	0.2000	8545588678	73660
4	59854	49678	2022-11-18	15.2000	3	0.2000	8545588678	73660
5	59854	88070	2022-11-18	15.2000	3	0.2000	8545588678	73660
6	59854	69302	2022-11-18	15.2000	3	0.2000	8545588678	73660
7	59854	43769	2022-11-18	18.4300	3	0.0300	924219421	74044
8	53028	85114	2022-11-05	32.6400	2	0.0400	1234567890	73660

Made with Streamlit





12. Conclusion

The project currently performs most basic CRUD operations on the database and comprises of triggers, functions, procedures, cursors etc. to help refine these CRUD operations. There is much more scope to this project and I will continue to build it going forward.

13. References

- [1] PESU Academy Slides and Notes
- [2] MySQL official documentation