SVKM's NMIMS

School of Technology Management & Engineering, Chandigarh

A.Y. 2023 - 24

Course: Database Management Systems

Project Report

Program	B.TECH AI & DS	
Semester	IV	
Name of the Project:	GROCERY STORE MANAGEMENT SYSTEM	
Details of Project Members		
Batch	Roll No.	Name
B1	A078	HARIPRIYA SARAF
B2	A104	LAKSHMEESH MANKAME
Date of Submission: 30/03/2024		

Contribution of each project Members:

Roll No.	Name:	Contribution
A104	LAKSHMEESH MANKAME	Database creation, SQL queries, Normalization, Report
A078	HARIPRIYA SARAF	ER Diagram, Normalization, Front end (GUI) creation, Report

Github link of your project: https://github.com/haripriyasaraf/Grocery-Store-Management-System/tree/main

Project Report

GROCERY STORE MANAGEMENT SYSTEM

By

LAKSHMEESH MANKAME, A104

HARIPRIYA SARAF, A078

Course: DBMS

AY: 2023-24

Table of Contents

I. Storyline

Introduction

Meet Sarah, the owner of FreshMart, a local grocery store struggling to keep track of inventory and manage supplier orders. Stockouts and frustrated customers are becoming a norm. Sarah knows she needs a change.

Enter, the FreshMart Management System! This innovative software promises to revolutionize Sarah's business. With features like real-time inventory tracking, automatic purchase orders, and expiry date monitoring, FreshMart can finally optimize its stock levels and reduce waste.

As Sarah implements the system, we see its impact. Shelf space is optimized, leading to a wider product variety. Automatic orders ensure popular items are always in stock, reducing out-of-stocks and customer frustration. Expiring items are identified early, allowing for discounts or donations, minimizing waste.

With data-driven insights, Sarah can now make informed decisions about promotions and restocking. FreshMart becomes a customer favorite, with its efficient service and wider selection. Sarah, relieved and happy, can finally focus on growing her business.

Our Grocery Store Management System is designed to streamline and automate the operations of a grocery store. The system aims to manage and maintain the product inventory, handle customer purchases, track orders and payments, manage employee details, and oversee advertising and supplier information.

With this system, the grocery store can efficiently manage its day-to-day operations, enhance customer experience, and optimize its business processes.

Product Management

The system includes a Products table to manage the store's inventory. Each product is uniquely identified and includes details such as Item_Name, Item_Price, GST, Discount_Offer, and Stock. This table categorizes products

into different categories. This enables the store to organize its inventory effectively and provide customers with a wide range of products to choose from.

Customer Management

The system maintains a Customers table to store customer information. This facilitates easy communication with customers and enables the store to provide personalized services and offers.

Purchase and Payment Management

The Buy table tracks the purchases made by customers, linking each purchase to a customer and a product. The Payment_Details table manages the payment information. The system also includes an Order table to manage customer orders, tracking numbers, and payment details.

Feedback and Rating System

To improve customer satisfaction and gather valuable feedback, the system includes a Feedback table where customers can rate products and provide comments. This helps the store to understand customer preferences, identify popular products, and make informed decisions on inventory management and product promotions.

Advertising and Supplier Management

The Advertises table manages the advertising of products by linking products to advertising companies through Company_Code. The system also maintains Supplier and Supplies tables to manage supplier information and the products supplied by them. This ensures timely replenishment of stock and maintains a good relationship with suppliers.

Employee and Admin Management

The system includes an Employee table to manage employee details. The Admin table stores the credentials of the system administrators to ensure secure access and management of the system.

Store Management

The Store table manages store details. The Store_Contact_No table stores the contact numbers of stores, enabling effective communication and coordination between the central office and individual stores.

II. Components of Database Design

Entities and Their Attributes:

1. Products
   - Item_Code (Primary Key)
   - Item_Name
   - Item_Price
   - GST
   - Discount_Offer
   - Stock

2. Product_Category
   - Category_Id (Primary Key)
   - Category_Name
   - Item_Code (Foreign Key)

3. Customers
   - Cust_Id (Primary Key)
   - Cust_Name
   - DOB
   - Email
   - Address

4. Customers_Contact_No
   - Contact_Number (Primary Key)
   - Cust_Id (Foreign Key)

5. Payment_Details
   - Payment_Id (Primary Key)
   - Payment_date
   - Total_Amount
   - Payment_Method
   - Emp_Code (Foreign Key)

6. Order
   - Order_Id (Primary Key)
   - Order_Date

- Total_Amount
- Tracking_Number
- Payment_Id (Foreign Key)
- Cust_Id (Foreign Key)

7. Bill
   - Order_Det_Id (Primary Key)
   - Date_Of_billing
   - Total_amount
   - Item_Code (Foreign Key)
   - Bill_Number

8. Tracking_Details
   - Tracking_Number (Primary Key)
   - Courier_Name

9. Feedback_Comment
   - Comment
   - Feedback_Id (Primary Key)

10. Feedback
    - Cust_Id (Foreign Key)
    - Item_Code (Foreign Key)
    - Rating
    - Feedback_date

11. Advertises
    - Item_Code (Foreign Key)
    - Company_Code (Foreign Key)

12. Advertising_Company
    - Company_Code (Primary Key)
    - Company_Name

13. Supplier
    - Supplier_ID (Primary Key)
    - Supplier_name
    - Supplier_address

14. Supplies

- Supplier_Id (Foreign Key)
- Item_Code (Foreign Key)

15. Supplier_Contact
    - Supplier_contact (Primary Key)
    - Supplier_Id (Foreign Key)

16. Supplier_Email
    - Supplier_email (Primary Key)
    - Supplier_ID (Foreign Key)

17. Employee
    - Emp_code (Primary Key)
    - Emp_name
    - Date
    - Month
    - Year
    - Age
    - Position
    - Salary
    - Username

18. Admin
    - Username (Primary Key)
    - Password
    - Admin_ID
    - Admin_name

19. Store
    - Store_ID (Primary Key)
    - Store_Name
    - Location
    - Operating_hrs
    - Username

20. Store_Contact_No
    - Contact_Number (Primary Key)
    - Store_ID (Foreign Key)

Relationships, Cardinality, and Participation:

1. Products - Product_Category
   - Relationship: One-to-Many
   - Cardinality: One Product can belong to one Product_Category, but one Product_Category can have many Products.
   - Participation: Mandatory on both sides.

2. Customers - Customers_Contact_No
   - Relationship: One-to-Many
   - Cardinality: One Customer can have many Contact_Numbers, but one Contact_Number belongs to only one Customer.
   - Participation: Mandatory on both sides.

3. Customers - Buy
   - Relationship: One-to-Many
   - Cardinality: One Customer can make many Purchases, but one Purchase is made by one Customer.
   - Participation: Mandatory on both sides.

4. Products - Buy
   - Relationship: One-to-Many
   - Cardinality: One Product can be bought in many Purchases, but one Purchase consists of one Product.
   - Participation: Mandatory on both sides.

5. Customers - Order
   - Relationship: One-to-Many
   - Cardinality: One Customer can place many Orders, but one Order is placed by one Customer.
   - Participation: Mandatory on both sides.

6. Payment_Details - Order
   - Relationship: One-to-One
   - Cardinality: One Order can have one Payment_Detail, and one Payment_Detail is associated with one Order.
   - Participation: Mandatory on both sides.

7. Products - Bill
   - Relationship: One-to-Many
   - Cardinality: One Product can be billed in many Bills, but one Bill consists of one Product.
   - Participation: Mandatory on both sides.

8. Order - Bill
   - Relationship: One-to-One
   - Cardinality: One Order can have one Bill, and one Bill is

 associated with one Order.
   - Participation: Mandatory on both sides.

9. Products - Feedback
   - Relationship: One-to-Many
   - Cardinality: One Product can have many Feedbacks, but one Feedback is given for one Product.
   - Participation: Mandatory on both sides.

10. Customers - Feedback
    - Relationship: One-to-Many
    - Cardinality: One Customer can give many Feedbacks, but one Feedback is given by one Customer.
    - Participation: Mandatory on both sides.

11. Products - Advertises
    - Relationship: Many-to-Many
    - Cardinality: One Product can be advertised by many Advertising_Companies, and one Advertising_Company can advertise many Products.
    - Participation: Mandatory on both sides.

12. Advertising_Company - Advertises
    - Relationship: Many-to-Many
    - Cardinality: One Advertising_Company can advertise many Products, and one Product can be advertised by many Advertising_Companies.
    - Participation: Mandatory on both sides.

13. Supplier - Supplies
    - Relationship: One-to-Many

- Cardinality: One Supplier can supply many Products, but one Product is supplied by one Supplier.
- Participation: Mandatory on both sides.

14. Products - Supplies
- Relationship: One-to-Many
- Cardinality: One Product can be supplied by many Suppliers, but one Supplier supplies one Product.
- Participation: Mandatory on both sides.

15. Supplier - Supplier_Contact
- Relationship: One-to-Many
- Cardinality: One Supplier can have many Contact_Numbers, but one Contact_Number belongs to only one Supplier.
- Participation: Mandatory on both sides.

16. Supplier - Supplier_Email
- Relationship: One-to-Many
- Cardinality: One Supplier can have many Emails, but one Email belongs to only one Supplier.
- Participation: Mandatory on both sides.

17. Employee - Payment_Details
- Relationship: One-to-Many
- Cardinality: One Employee can process many Payments, but one Payment is processed by one Employee.
- Participation: Mandatory on both sides.

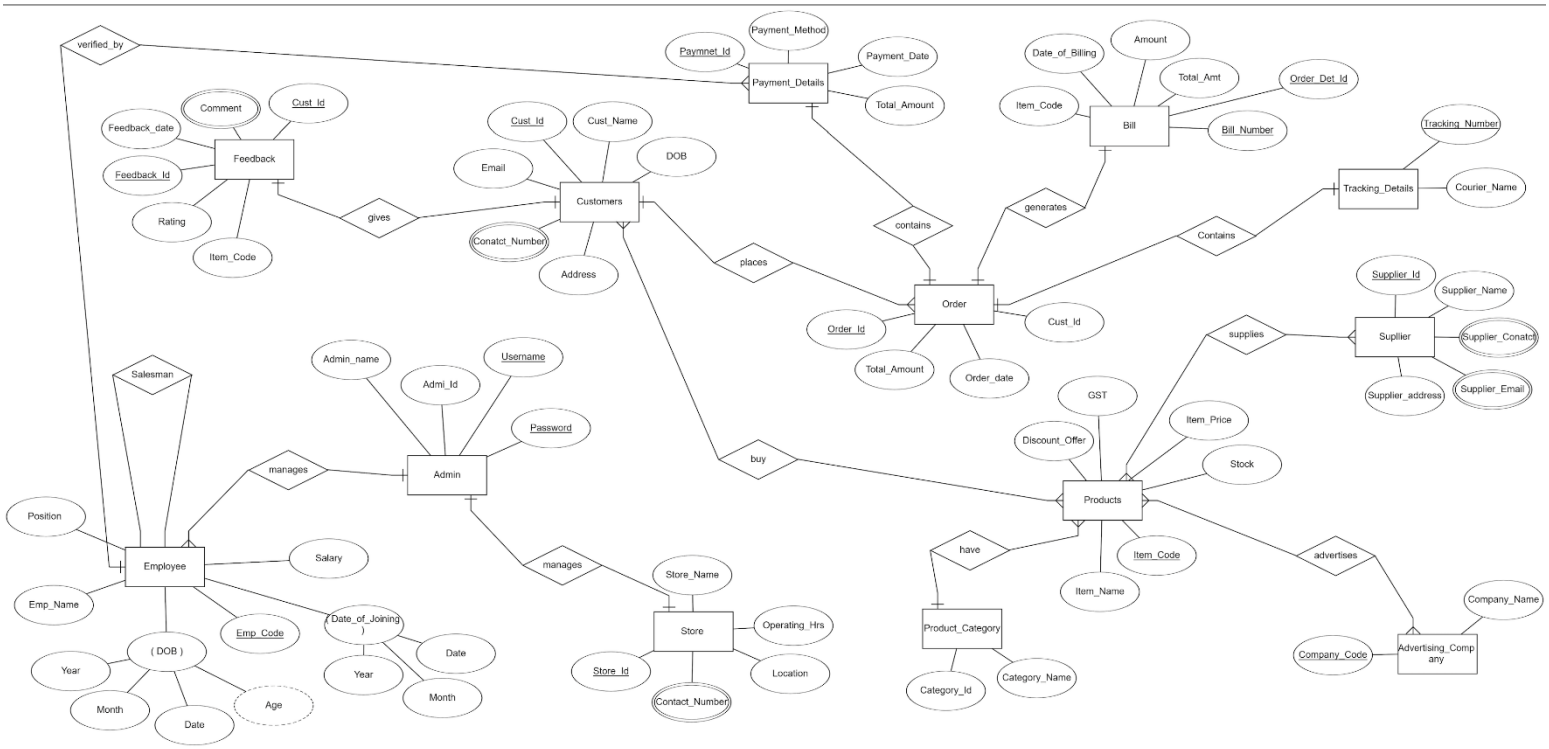
18. Employee - Store
- Relationship: One-to-Many
- Cardinality: One Employee can manage many Stores, but one Store is managed by one Employee.
- Participation: Mandatory on both sides.

19. Store - Store_Contact_No
- Relationship: One-to-Many
- Cardinality: One Store can have many Contact_Numbers, but one Contact_Number belongs to only one Store.
- Participation: Mandatory on both sides.

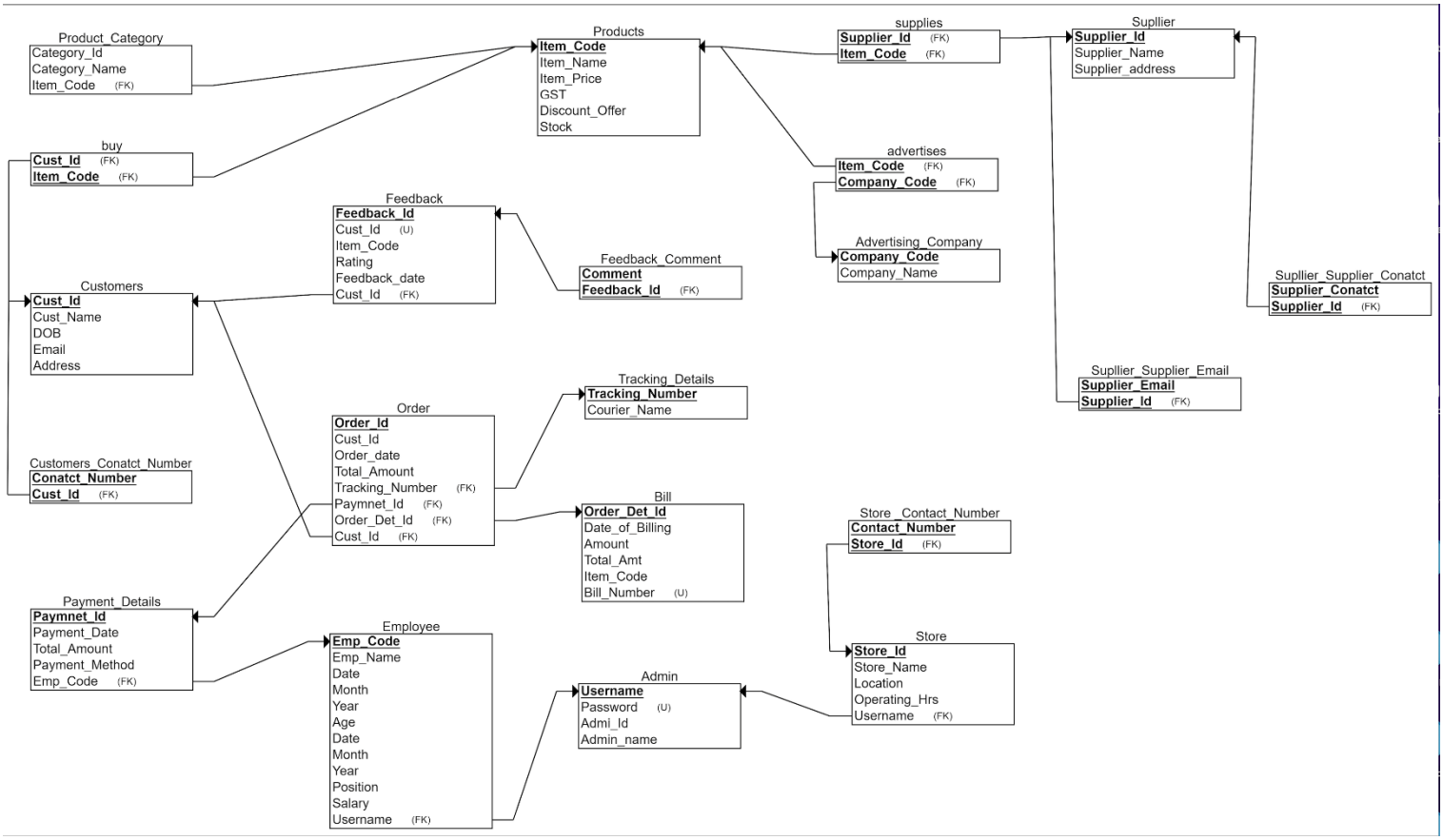
20. Admin - Employee
- Relationship: One-to-Many
- Cardinality: One Admin can manage many Employees, but one Employee is managed by one Admin.
- Participation: Mandatory on both sides.

21. Admin - Store
- Relationship: One-to-Many
- Cardinality: One Admin can manage many Stores, but one Store is managed by one Admin.
- Participation: Mandatory on both sides.

III. Entity Relationship Diagram



IV. Relational Model



V. Normalization

Normalization:-

Products(Item_Code , Item_Name, Item_Price , GST, Discount_Offer , Stock)
Item_Code -> Item_Name , Item_Price , GST , Discount_Offer ,Stock
Item_Name -> Item_Code , Item_Price , GST , Discount_Offer ,Stock
Prime attributes :- Item_Code , Item_Name
Non Prime Attributes :- Item_Price , GST , Discount_Offer ,Stock
Candidate Key :- Item_Code , Item_Name
1NF - Table has a primary key as well as 2 candidate keys and there are no multivalued or composite attributes. Hence 1NF.
2NF - Since only prime attributes can derive non prime attributes , thus above table is in 2NF.
3NF - Since there is no transitive dependency we can say that above table is in 3NF.

Product_Category(Category_Id , Category_Name , Item_Code)
Category_Id -> Category_Name
Item_Code -> Category_Id , Category_Name
Prime attribute :- Item_Code
Non Prime Attribute :- Category_Id , Category_Name
1NF - all attributes are atomic as well as we have primary key hence it is in 1NF.
2NF - only the prime attribute is deriving all other attributes hence 2NF.
3NF - there is no transitive dependency hence 3NF.

Buy(Cust_Id , Item_Code)
1NF - since all attributes are atomic hence it is in 1NF.
2NF - since there is no partial dependency thus it is in 2NF
3NF - no transitive dependency hence 3NF

Customers(Cust_Id , Cust_Name, DOB, Email, Address)
Cust_Id -> Cust_Name, DOB, Email, Address
Email -> Cust_Id , Cust_Name, DOB, Email, Address
Prime attributes :- Cust_Id, Email
Non Prime attributes :- DOB, Cust_Name, Address
1NF - all attributes are atomic
2NF - no partial dependency
3NF - No transitive dependency

Customers_Contact_No(Contact_Number , Cust_Id)
1NF - all attributes are atomic
2NF - No partial dependency
3NF - No transitive dependency

Payment_Details(Payment_Id , Payment_date, Total_Amount, Payment_Method, Emp_Code)
Payment_Id -> Payment_date , Total_Amount , Payment_Method, Emp_Code
1NF - all attributes are atomic
2NF - no partial dependency
3NF - No transitive dependency

Order(Order_Id , Order_Date, Total_Amount, Tracking_Number , Payment_Id , Order_Det_Id , Cust_Id)
Order_Id -> Order_Date , Total_Amount , Tracking_Number , Payment_Id, Order_Det_Id, Cust_Id
Tracking_Number -> Order_Date , Total_Amount , Payment_Id, Order_Det_Id, Cust_Id, Order_Id
Payment_Id -> Order_Date , Total_Amount , Order_Det_Id, Cust_Id, Order_Id, Tracking_Number
Order_Det_Id -> Order_Date , Total_Amount , Payment_Id, Cust_Id, Order_Id, Tracking_Number
Prime attributes :- Order_Id, Tracking_Nmuber , Payment_Id, Order_Det_Id
Non Prime Attributes :- Order_Date , Total_Amount , Cust_Id
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Bill(Order_Det_Id , Date_Of_billing, Total_amount, Item_Code , Bill_Number)
Order_Det_Id -> Date_Of_Billing , Total_amount, Item_Code, Bill_Number
Bill_Number -> Date_Of_Billing , Total_amount, Item_Code, Order_Det_Id
Prime attributes :- Order_Det_Id , Bill_Number
Non prime attributes :- Date_Of_Billing , Total_amount, Item_Code
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Tracking_Details(Tracking_Number , Courier_Name)
Tracking_Number -> Courier_Name
Prime attribute :- Tracking_Number
Non - Prime Attribute :- Courier_Name
Candidate key / Primary Key :- Tracking_Number

1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Feedback_Comment(Comment , Feedback_Id)
Feedback_Id -> Comment
Prime attribute :- Feedback_Id
Non prime attribute :- Comment
Candidate key / Primary Key :- Feedback_Id
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Feedback(Feedback_Id , Cust_Id , Item_Code , Rating, Feedback_date)
Feedback_Id -> Cust_Id , Item_Code , Rating, Feedback_Date
Cust_Id -> Item_Code , Rating, Feedback_Date , Feedback_Id
Item_Code -> Cust_Id, Rating, Feedback_Date , Feedback_Id
Prime_Attribute :- Feedback_Id , Cust_Id , Item_Code
Non prime attribute :- Rating, Feedback_Date
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Advertises(Item_Code,Company_Code)
Company_Code -> Item_Code
Item_Code -> Company_Code
Prime attributes :- Company_Code , Item_Code
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Advertising_Company(Company_Code, Company_Name)
Company_Code -> Comapny_Name
Prime Attribute :- Comapny_Code
Non prime Attribute :- Comapny_Name
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Supplier(Supplier_ID,Supplier_name,Supplier_address)
Supplier_Id -> Supplier_Name , Supplier_address
1NF - all attributes are atomic
2NF - no partial dependency

3NF - no transitive dependency

Supplies(Supplier_Id , Item_Code)
Supplier_Id -> Item-Code
Item_Code -> Supplier_Id
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Supplier_Contact(Supplier_contact, Supplier_Id)
Supplier_Id -> Supplier_Contact
Prime attribute :- Supplier_Id
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Supplier_Email(Supplier_email, Supplier_ID)
Supplier_Id -> Supplier_Email
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Employee(Emp_code, Emp_name, Date, Month, Year, Age, Position, Salary, Username)
Emp_Code -> Emp_Name , Date, Month, Year, Age, Position, Salary, Username
Username -> Emp_Code , Emp_Name , Date, Month, Year, Age, Position, Salary
Prime attributes :- Emp_Code , Username
Non Prime attributes :- Emp_Name , Date, Month, Year, Age, Position, Salary
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Admin(Username,Password,Admin_ID,Admin_name)
Username -> Password, Admin_Id , Admin_Name
Admin_Id -> Username, password, Admin_Name
Prime attributes :- Username , Admin_Id
Non prime attributes :- Admin_Name, Password
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Store(Store_ID,Store_Name,Location,Operating_hrs)
Store_Id -> Store_Name , Location , Operatong_Hours
Prime attributes :- Store_Id
Non prime attributes :- Store_Name , Location , Operatong_Hours
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

Store_Contact_No(Contact_Number, Store_ID)
Store_Id -> Conatct_Number
1NF - all attributes are atomic
2NF - no partial dependency
3NF - no transitive dependency

VI. SQL Queries

1) CREATING TABLES:

```sql
create database project;
use project;
CREATE TABLE Products (
    Item_Code INTEGER PRIMARY KEY,
    Item_Name TEXT,
    Item_Price REAL,
    GST REAL,
    Discount_Offer REAL,
    Stock INTEGER
);

CREATE TABLE Product_Category (
    Category_Id INTEGER PRIMARY KEY,
    Category_Name TEXT,
    Item_Code INTEGER,
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);

CREATE TABLE Buy (
    Cust_Id INTEGER,
    Item_Code INTEGER,
    PRIMARY KEY (Cust_Id, Item_Code),
    FOREIGN KEY (Cust_Id) REFERENCES Customers(Cust_Id),
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);
CREATE TABLE Customers (
    Cust_Id INTEGER PRIMARY KEY,
    Cust_Name TEXT,
    DOB TEXT,
    Email TEXT,
    Address TEXT
);

CREATE TABLE Customers_Contact_No (
    Contact_Number INTEGER PRIMARY KEY,
    Cust_Id INTEGER,
    FOREIGN KEY (Cust_Id) REFERENCES Customers(Cust_Id)
);

CREATE TABLE Payment_Details (
    Payment_Id INTEGER PRIMARY KEY,
    Payment_date TEXT,
    Total_Amount REAL,
    Payment_Method TEXT,
    Emp_Code INTEGER,
    FOREIGN KEY (Emp_Code) REFERENCES Employee(Emp_code)
);

```

```sql
CREATE TABLE Order1(
    Order_Id INTEGER PRIMARY KEY,
    Order_Date TEXT,
    Total_Amount REAL,
    Tracking_Number TEXT,
    Payment_Id INTEGER,
    Order_Det_Id INTEGER,
    Cust_Id INTEGER,
    FOREIGN KEY (Payment_Id) REFERENCES Payment_Details(Payment_Id),
    FOREIGN KEY (Cust_Id) REFERENCES Customers(Cust_Id)
);

CREATE TABLE Bill (
    Order_Det_Id INTEGER PRIMARY KEY,
    Date_Of_billing TEXT,
    Total_amount REAL,
    Item_Code INTEGER,
    Bill_Number INTEGER,
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);

CREATE TABLE Tracking_Details (
    Tracking_Number varchar(255) PRIMARY KEY,
    Courier_Name VARCHAR(255)
);

CREATE TABLE Feedback_Comment (
    Comment varchar(255),
    Feedback_Id INTEGER PRIMARY KEY
);

CREATE TABLE Customer_Feedback (
    Cust_Id INTEGER,
    Item_Code INTEGER,
    Rating INTEGER,
    Feedback_date TEXT,
    PRIMARY KEY (Cust_Id, Item_Code),
    FOREIGN KEY (Cust_Id) REFERENCES Customers(Cust_Id),
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);


CREATE TABLE Advertises (
    Item_Code INTEGER,
    Company_Code INTEGER,
    PRIMARY KEY (Item_Code, Company_Code),
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);

CREATE TABLE Advertising_Company (
    Company_Code INTEGER PRIMARY KEY,
    Company_Name TEXT
);

CREATE TABLE Supplier (
    Supplier_ID INTEGER PRIMARY KEY,
    Supplier_name TEXT,
    Supplier_address TEXT
);

CREATE TABLE Supplies (
    Supplier_Id INTEGER,
    Item_Code INTEGER,
    PRIMARY KEY (Supplier_Id, Item_Code),
    FOREIGN KEY (Supplier_Id) REFERENCES Supplier(Supplier_ID),
    FOREIGN KEY (Item_Code) REFERENCES Products(Item_Code)
);

CREATE TABLE Supplier_Contact (
    Supplier_contact INTEGER PRIMARY KEY,
    Supplier_Id INTEGER,
    FOREIGN KEY (Supplier_Id) REFERENCES Supplier(Supplier_ID)
);

CREATE TABLE Supplier_Email (
    Supplier_email varchar(255) PRIMARY KEY,
    Supplier_ID INTEGER,
    FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)
);
```

```sql
CREATE TABLE Employee (
    Emp_code INTEGER PRIMARY KEY,
    Emp_name TEXT,
    Date TEXT,
    Month TEXT,
    Year TEXT,
    Age INTEGER,
    Position TEXT,
    Salary REAL,
    Username TEXT
);

CREATE TABLE Admin (
    Username varchar(255) PRIMARY KEY,
    Password varchar(255),
    Admin_ID INTEGER,
    Admin_name varchar(255)
);

CREATE TABLE Store (
    Store_ID INTEGER PRIMARY KEY,
    Store_Name varchar(255),
    Location varchar(255),
    Operating_hrs varchar(255),
    Username varchar(255)
);

CREATE TABLE Store_Contact_No (
    Contact_Number INTEGER PRIMARY KEY,
    Store_ID INTEGER,
    FOREIGN KEY (Store_ID) REFERENCES Store(Store_ID)
);
```

2) INSERTING VALUES INTO TABLE:

```sql
-- Products
INSERT INTO Products VALUES (1, 'Milk', 2.5, 0.05, 0, 100);
INSERT INTO Products VALUES (2, 'Bread', 1.5, 0.05, 0, 150);
INSERT INTO Products VALUES (3, 'Apple', 0.5, 0.05, 0, 200);
INSERT INTO Products VALUES (4, 'Orange Juice', 3, 0.1, 0, 80);
INSERT INTO Products VALUES (5, 'Chocolate', 2, 0.05, 0.1, 120);
INSERT INTO Products VALUES (6, 'Rice', 10, 0.1, 0, 50);
INSERT INTO Products VALUES (7, 'Chicken', 8, 0.15, 0, 30);
INSERT INTO Products VALUES (8, 'Tomato', 0.3, 0.05, 0, 220);
INSERT INTO Products VALUES (9, 'Potato', 0.2, 0.05, 0, 300);
INSERT INTO Products VALUES (10, 'Eggs', 1, 0.05, 0, 100);

-- Product_Category
INSERT INTO Product_Category VALUES (1, 'Dairy', 1);
INSERT INTO Product_Category VALUES (2, 'Bakery', 2);
INSERT INTO Product_Category VALUES (3, 'Fruits', 3);
INSERT INTO Product_Category VALUES (4, 'Beverages', 4);
INSERT INTO Product_Category VALUES (5, 'Sweets', 5);
INSERT INTO Product_Category VALUES (6, 'Grains', 6);
INSERT INTO Product_Category VALUES (7, 'Meat', 7);
INSERT INTO Product_Category VALUES (8, 'Vegetables', 8);
INSERT INTO Product_Category VALUES (9, 'Vegetables', 9);
INSERT INTO Product_Category VALUES (10, 'Dairy', 10);
```

```sql
-- Customers
INSERT INTO Customers VALUES (1, 'John Doe', '1990-05-15', 'johndoe@email.com', '123 Main St');
INSERT INTO Customers VALUES (2, 'Jane Smith', '1985-07-20', 'janesmith@email.com', '456 Elm St');
INSERT INTO Customers VALUES (3, 'Alice Johnson', '1992-02-10', 'alice@email.com', '789 Oak St');
INSERT INTO Customers VALUES (4, 'Bob Williams', '1988-09-05', 'bob@email.com', '101 Pine St');
INSERT INTO Customers VALUES (5, 'Charlie Brown', '1995-04-18', 'charlie@email.com', '202 Cedar St');
INSERT INTO Customers VALUES (6, 'David Davis', '1993-11-12', 'david@email.com', '303 Maple St');
INSERT INTO Customers VALUES (7, 'Emily Jones', '1991-08-25', 'emily@email.com', '404 Birch St');
INSERT INTO Customers VALUES (8, 'Frank White', '1987-03-30', 'frank@email.com', '505 Spruce St');
INSERT INTO Customers VALUES (9, 'Grace Lee', '1994-06-22', 'grace@email.com', '606 Pineapple St');
INSERT INTO Customers VALUES (10, 'Hannah Clark', '1996-01-05', 'hannah@email.com', '707 Mango St');

-- Customers_Contact_No
INSERT INTO Customers_Contact_No VALUES (1234567890, 1);
INSERT INTO Customers_Contact_No VALUES (1234567891, 2);
INSERT INTO Customers_Contact_No VALUES (1234567892, 3);
INSERT INTO Customers_Contact_No VALUES (1234567893, 4);
INSERT INTO Customers_Contact_No VALUES (1234567894, 5);
INSERT INTO Customers_Contact_No VALUES (1234567895, 6);
INSERT INTO Customers_Contact_No VALUES (1234567896, 7);
INSERT INTO Customers_Contact_No VALUES (1234567897, 8);
INSERT INTO Customers_Contact_No VALUES (1234567898, 9);
INSERT INTO Customers_Contact_No VALUES (1234567899, 10);

-- Payment_Details
INSERT INTO Payment_Details VALUES (1, '2024-03-28', 25, 'Credit Card', 1);
INSERT INTO Payment_Details VALUES (2, '2024-03-29', 15, 'Debit Card', 2);
INSERT INTO Payment_Details VALUES (3, '2024-03-30', 5, 'Cash', 3);
INSERT INTO Payment_Details VALUES (4, '2024-03-31', 20, 'Credit Card', 4);
INSERT INTO Payment_Details VALUES (5, '2024-04-01', 10, 'Debit Card', 5);
INSERT INTO Payment_Details VALUES (6, '2024-04-02', 30, 'Cash', 6);
INSERT INTO Payment_Details VALUES (7, '2024-04-03', 40, 'Credit Card', 7);
INSERT INTO Payment_Details VALUES (8, '2024-04-04', 35, 'Debit Card', 8);
INSERT INTO Payment_Details VALUES (9, '2024-04-05', 45, 'Cash', 9);
INSERT INTO Payment_Details VALUES (10, '2024-04-06', 50, 'Credit Card', 10);
```

```sql
-- Order
INSERT INTO Order1 VALUES (1, '2024-03-28', 25, 'TN12345', 1, 1, 1);
INSERT INTO Order1 VALUES (2, '2024-03-29', 15, 'TN12346', 2, 2, 2);
INSERT INTO Order1 VALUES (3, '2024-03-30', 5, 'TN12347', 3, 3, 3);
INSERT INTO Order1 VALUES (4, '2024-03-31', 20, 'TN12348', 4, 4, 4);
INSERT INTO Order1 VALUES (5, '2024-04-01', 10, 'TN12349', 5, 5, 5);
INSERT INTO Order1 VALUES (6, '2024-04-02', 30, 'TN12350', 6, 6, 6);
INSERT INTO Order1 VALUES (7, '2024-04-03', 40, 'TN12351', 7, 7, 7);
INSERT INTO Order1 VALUES (8, '2024-04-04', 35, 'TN12352', 8, 8, 8);
INSERT INTO Order1 VALUES (9, '2024-04-05', 45, 'TN12353', 9, 9, 9);
INSERT INTO Order1 VALUES (10, '2024-04-06', 50, 'TN12354', 10, 10, 10);

-- Bill
INSERT INTO Bill VALUES (1, '2024-03-28', 25, 1, 101);
INSERT INTO Bill VALUES (2, '2024-03-29', 15, 2, 102);
INSERT INTO Bill VALUES (3, '2024-03-30', 5, 3, 103);
INSERT INTO Bill VALUES (4, '2024-03-31', 20, 4, 104);
INSERT INTO Bill VALUES (5, '2024-04-01', 10, 5, 105);
INSERT INTO Bill VALUES (6, '2024-04-02', 30, 6, 106);
INSERT INTO Bill VALUES (7, '2024-04-03', 40, 7, 107);
INSERT INTO Bill VALUES (8, '2024-04-04', 35, 8, 108);
INSERT INTO Bill VALUES (9, '2024-04-05', 45, 9, 109);
INSERT INTO Bill VALUES (10, '2024-04-06', 50, 10, 110);

-- Tracking_Details
INSERT INTO Tracking_Details VALUES ('TN12345', 'FastCourier');
INSERT INTO Tracking_Details VALUES ('TN12346', 'QuickShip');
INSERT INTO Tracking_Details VALUES ('TN12347', 'SpeedyDelivery');
INSERT INTO Tracking_Details VALUES ('TN12348', 'ExpressDelivery');
INSERT INTO Tracking_Details VALUES ('TN12349', 'SwiftCourier');
INSERT INTO Tracking_Details VALUES ('TN12350', 'RapidShip');
INSERT INTO Tracking_Details VALUES ('TN12351', 'ZoomDelivery');
INSERT INTO Tracking_Details VALUES ('TN12352', 'SpeedyShip');
INSERT INTO Tracking_Details VALUES ('TN12353', 'LightningCourier');
INSERT INTO Tracking_Details VALUES ('TN12354', 'FlashDelivery');
```

```sql
-- Advertising_Company
INSERT INTO Advertising_Company VALUES (1, 'FreshAds');
INSERT INTO Advertising_Company VALUES (2, 'QuickAds');
INSERT INTO Advertising_Company VALUES (3, 'FruitAds');
INSERT INTO Advertising_Company VALUES (4, 'DrinkAds');
INSERT INTO Advertising_Company VALUES (5, 'SweetAds');
INSERT INTO Advertising_Company VALUES (6, 'GrainAds');
INSERT INTO Advertising_Company VALUES (7, 'MeatAds');
INSERT INTO Advertising_Company VALUES (8, 'VeggieAds');
INSERT INTO Advertising_Company VALUES (9, 'TaterAds');
INSERT INTO Advertising_Company VALUES (10, 'EggAds');

-- Supplier
INSERT INTO Supplier VALUES (1, 'MilkSupplier', '123 Dairy St');
INSERT INTO Supplier VALUES (2, 'BreadSupplier', '456 Bakery St');
INSERT INTO Supplier VALUES (3, 'AppleSupplier', '789 Orchard St');
INSERT INTO Supplier VALUES (4, 'JuiceSupplier', '101 Beverage St');
INSERT INTO Supplier VALUES (5, 'ChocolateSupplier', '202 Sweets St');
INSERT INTO Supplier VALUES (6, 'RiceSupplier', '303 Grain St');
INSERT INTO Supplier VALUES (7, 'ChickenSupplier', '404 Meat St');
INSERT INTO Supplier VALUES (8, 'TomatoSupplier', '505 Veggie St');
INSERT INTO Supplier VALUES (9, 'PotatoSupplier', '606 Spud St');
INSERT INTO Supplier VALUES (10, 'EggSupplier', '707 Farm St');

-- Supplies
INSERT INTO Supplies VALUES (1, 1);
INSERT INTO Supplies VALUES (2, 2);
INSERT INTO Supplies VALUES (3, 3);
INSERT INTO Supplies VALUES (4, 4);
INSERT INTO Supplies VALUES (5, 5);
INSERT INTO Supplies VALUES (6, 6);
INSERT INTO Supplies VALUES (7, 7);
INSERT INTO Supplies VALUES (8, 8);
INSERT INTO Supplies VALUES (9, 9);
INSERT INTO Supplies VALUES (10, 10);

-- Supplier_Contact
INSERT INTO Supplier_Contact VALUES (1111111111, 1);
INSERT INTO Supplier_Contact VALUES (1111111112, 2);
INSERT INTO Supplier_Contact VALUES (1111111113, 3);
INSERT INTO Supplier_Contact VALUES (1111111114, 4);
INSERT INTO Supplier_Contact VALUES (1111111115, 5);
INSERT INTO Supplier_Contact VALUES (1111111116, 6);
INSERT INTO Supplier_Contact VALUES (1111111117, 7);
INSERT INTO Supplier_Contact VALUES (1111111118, 8);
INSERT INTO Supplier_Contact VALUES (1111111119, 9);
INSERT INTO Supplier_Contact VALUES (1111111120, 10);

-- Supplier_Email
INSERT INTO Supplier_Email VALUES ('milk@email.com', 1);
INSERT INTO Supplier_Email VALUES ('bread@email.com', 2);
INSERT INTO Supplier_Email VALUES ('apple@email.com', 3);
INSERT INTO Supplier_Email VALUES ('juice@email.com', 4);
INSERT INTO Supplier_Email VALUES ('chocolate@email.com', 5);
INSERT INTO Supplier_Email VALUES ('rice@email.com', 6);
INSERT INTO Supplier_Email VALUES ('chicken@email.com', 7);
INSERT INTO Supplier_Email VALUES ('tomato@email.com', 8);
INSERT INTO Supplier_Email VALUES ('potato@email.com', 9);
INSERT INTO Supplier_Email VALUES ('egg@email.com', 10);

-- Employee
INSERT INTO Employee VALUES (1, 'Alex', '2024-01-01', 'January', '2024', 30, 'Manager', 5000, 'alex123');
INSERT INTO Employee VALUES (2, 'Brian', '2024-02-02', 'February', '2024', 25, 'Salesperson', 3000, 'brian123');
INSERT INTO Employee VALUES (3, 'Charlie', '2024-03-03', 'March', '2024', 22, 'Cashier', 2500, 'charlie123');
INSERT INTO Employee VALUES (4, 'David', '2024-04-04', 'April', '2024', 28, 'Storekeeper', 3200, 'david123');
INSERT INTO Employee VALUES (5, 'Emma', '2024-05-05', 'May', '2024', 35, 'Manager', 5500, 'emma123');
INSERT INTO Employee VALUES (6, 'Frank', '2024-06-06', 'June', '2024', 26, 'Salesperson', 3100, 'frank123');
INSERT INTO Employee VALUES (7, 'Grace', '2024-07-07', 'July', '2024', 23, 'Cashier', 2600, 'grace123');
INSERT INTO Employee VALUES (8, 'Henry', '2024-08-08', 'August', '2024', 29, 'Storekeeper', 3300, 'henry123');
INSERT INTO Employee VALUES (9, 'Ivy', '2024-09-09', 'September', '2024', 32, 'Manager', 5200, 'ivy123');
INSERT INTO Employee VALUES (10, 'Jack', '2024-10-10', 'October', '2024', 27, 'Salesperson', 2900, 'jack123');
```

```sql
-- Admin
INSERT INTO Admin VALUES ('admin1', 'password1', 1, 'John');
INSERT INTO Admin VALUES ('admin2', 'password2', 2, 'Jane');
INSERT INTO Admin VALUES ('admin3', 'password3', 3, 'Alice');
INSERT INTO Admin VALUES ('admin4', 'password4', 4, 'Bob');
INSERT INTO Admin VALUES ('admin5', 'password5', 5, 'Charlie');
INSERT INTO Admin VALUES ('admin6', 'password6', 6, 'David');
INSERT INTO Admin VALUES ('admin7', 'password7', 7, 'Emily');
INSERT INTO Admin VALUES ('admin8', 'password8', 8, 'Frank');
INSERT INTO Admin VALUES ('admin9', 'password9', 9, 'Grace');
INSERT INTO Admin VALUES ('admin10', 'password10', 10, 'Henry');

-- Store
INSERT INTO Store VALUES (1, 'StoreA', 'New York', '9am-9pm', 'alex123');
INSERT INTO Store VALUES (2, 'StoreB', 'Los Angeles', '8am-10pm', 'brian123');
INSERT INTO Store VALUES (3, 'StoreC', 'Chicago', '10am-8pm', 'charlie123');
INSERT INTO Store VALUES (4, 'StoreD', 'Houston', '9am-9pm', 'david123');
INSERT INTO Store VALUES (5, 'StoreE', 'Phoenix', '8am-10pm', 'emma123');
INSERT INTO Store VALUES (6, 'StoreF', 'Philadelphia', '10am-8pm', 'frank123');
INSERT INTO Store VALUES (7, 'StoreG', 'San Antonio', '9am-9pm', 'grace123');
INSERT INTO Store VALUES (8, 'StoreH', 'San Diego', '8am-10pm', 'henry123');
INSERT INTO Store VALUES (9, 'StoreI', 'Dallas', '10am-8pm', 'ivy123');
INSERT INTO Store VALUES (10, 'StoreJ', 'San Jose', '9am-9pm', 'jack123');

-- Store_Contact_No
INSERT INTO Store_Contact_No VALUES (1111111100, 1);
INSERT INTO Store_Contact_No VALUES (1111111101, 2);
INSERT INTO Store_Contact_No VALUES (1111111102, 3);
INSERT INTO Store_Contact_No VALUES (1111111103, 4);
INSERT INTO Store_Contact_No VALUES (1111111104, 5);
INSERT INTO Store_Contact_No VALUES (1111111105, 6);
INSERT INTO Store_Contact_No VALUES (1111111106, 7);
INSERT INTO Store_Contact_No VALUES (1111111107, 8);
INSERT INTO Store_Contact_No VALUES (1111111108, 9);
INSERT INTO Store_Contact_No VALUES (1111111109, 10);
```

3) SQL QUERIES

1) Retrieve all product names along with their categories.

CODE:

```sql
SELECT Products.Item_Name, Product_Category.Category_Name
FROM Products
INNER JOIN Product_Category ON Products.Item_Code = Product_Category.Item_Code;
```

OUTPUT:

Item_Name	Category_Name
Milk	Dairy
Bread	Bakery
Apple	Fruits
Orange Juice	Beverages
Chocolate	Sweets
Rice	Grains
Chicken	Meat
Tomato	Vegetables
Potato	Vegetables
Eggs	Dairy

2) Retrieve the total amount and payment method for each order.
CODE:

```
SELECT Order1.Order_Id, Payment_Details.Total_Amount, Payment_Details.Payment_Method
FROM Order1
INNER JOIN Payment_Details ON Order1.Payment_Id = Payment_Details.Payment_Id;
```

OUTPUT:

Order_Id	Total_Amount	Payment_Method
1	25	Credit Card
2	15	Debit Card
3	5	Cash
4	20	Credit Card
5	10	Debit Card
6	30	Cash
7	40	Credit Card
8	35	Debit Card
9	45	Cash
10	50	Credit Card

3) Retrieve the total stock for each product category.
CODE:

```
SELECT Product_Category.Category_Name, SUM(Products.Stock) AS Total_Stock
FROM Product_Category
INNER JOIN Products ON Product_Category.Item_Code = Products.Item_Code
GROUP BY Product_Category.Category_Name;
```

OUTPUT:

Category_Name	Total_Stock
Dairy	200
Bakery	150
Fruits	200
Beverages	80
Sweets	120
Grains	50
Meat	30
Vegetables	520

4) Retrieve all orders placed by customers with their feedback ratings.
CODE:

```
SELECT Order1.Order_Id, Customer_Feedback.Rating
FROM Order1
INNER JOIN Customer_Feedback ON Order1.Cust_Id = Customer_Feedback.Cust_Id;
```

OUTPUT:

Order_Id	Rating
1	5
2	4
3	5
4	4
5	5
6	4
7	5
8	4
9	5
10	4

5) Retrieve the total amount of sales for each product.

CODE:

```
SELECT Products.Item_Name, SUM(Bill.Total_amount) AS Total_Sales
FROM Products
INNER JOIN Bill ON Products.Item_Code = Bill.Item_Code
GROUP BY Products.Item_Name;
```

OUTPUT:

Item_Name	Total_Sales
Milk	25
Bread	15
Apple	5
Orange Juice	20
Chocolate	10
Rice	30
Chicken	40
Tomato	35
Potato	45
Eggs	50

6) Retrieve all suppliers who supply 'Milk' and their contact details.

CODE:

```
SELECT Supplier.Supplier_name, Supplier_Contact.Supplier_contact, Supplier_Email.Supplier_email
FROM Supplier
INNER JOIN Supplies ON Supplier.Supplier_ID = Supplies.Supplier_Id
INNER JOIN Products ON Supplies.Item_Code = Products.Item_Code
INNER JOIN Supplier_Contact ON Supplier.Supplier_ID = Supplier_Contact.Supplier_Id
INNER JOIN Supplier_Email ON Supplier.Supplier_ID = Supplier_Email.Supplier_ID
WHERE Products.Item_Name = 'Milk';
```

OUTPUT:

	Supplier_name	Supplier_cont...	Supplier_email
	MilkSupplier	1111111111	milk@email.com

7) Retrieve the average rating of each product.
CODE:

```
SELECT Products.Item_Name, AVG(Customer_Feedback.Rating) AS Average_Rating
FROM Products
LEFT JOIN Customer_Feedback ON Products.Item_Code = Customer_Feedback.Item_Code
GROUP BY Products.Item_Name;
```

OUTPUT:

Item_Name	Average_Rating
Milk	5.0000
Bread	4.0000
Apple	5.0000
Orange Juice	4.0000
Chocolate	5.0000
Rice	4.0000
Chicken	5.0000
Tomato	4.0000
Potato	5.0000
Eggs	4.0000

8) Retrieve all payments made in cash greater than $30.
CODE:

```
SELECT *
FROM Payment_Details
WHERE Payment_Details.Payment_Method = 'Cash' AND Payment_Details.Total_Amount > 30;
```

OUTPUT:

	Payment_Id	Payment_date	Total_Amount	Payment_Method	Emp_Code
	9	2024-04-05	45	Cash	9
	NULL	NULL	NULL	NULL	NULL

9) Retrieve the name and total sales of the top 5 selling products.
CODE:

```
SELECT Products.Item_Name, SUM(Bill.Total_amount) AS Total_Sales
FROM Products
INNER JOIN Bill ON Products.Item_Code = Bill.Item_Code
GROUP BY Products.Item_Name
ORDER BY Total_Sales DESC
LIMIT 5;
```

OUTPUT:

Item_Name	Total_Sales
Eggs	50
Potato	45
Chicken	40
Tomato	35
Rice	30

10) Retrieve all customers who have not given any feedback.
CODE:

```
SELECT Customers.Cust_Name
FROM Customers
LEFT JOIN Customer_Feedback ON Customers.Cust_Id = Customer_Feedback.Cust_Id
WHERE Customer_Feedback.Cust_Id IS NULL;
```

OUTPUT:

Cust_Name

11) Retrieve the details of the youngest employees.
CODE:

```
SELECT * FROM Employee
ORDER BY Date, Month, Year
LIMIT 1;
```

OUTPUT:

Emp_code	Emp_name	Date	Month	Year	Age	Position	Salary	Username
1	Alex	2024-01-01	January	2024	30	Manager	5000	alex123
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

12) Retrieve the names of the advertising companies that advertise 'Bread'.

CODE:

```
SELECT Advertising_Company.Company_Name
FROM Advertising_Company
INNER JOIN Advertises ON Advertising_Company.Company_Code = Advertises.Company_Code
INNER JOIN Products ON Advertises.Item_Code = Products.Item_Code
WHERE Products.Item_Name = 'Bread';
```

OUTPUT:

Company_Name
QuickAds

13) Retrieve the total sales amount for each payment method.

CODE:

```
SELECT Payment_Details.Payment_Method, SUM(Order1.Total_Amount) AS Total_Sales
FROM Order1
INNER JOIN Payment_Details ON Order1.Payment_Id = Payment_Details.Payment_Id
GROUP BY Payment_Details.Payment_Method;
```

OUTPUT:

Payment_Method	Total_Sales
Credit Card	135
Debit Card	60
Cash	80

14) Retrieve all products with a discount offer greater than 10%.

CODE:

```
SELECT *
FROM Products
WHERE Products.Discount_Offer > 10;
```

OUTPUT:

Item_Code	Item_Name	Item_Price	GST	Discount_Off...	Stock
NULL	NULL	NULL	NULL	NULL	NULL

15) Retrieve the name of the courier for each order.

CODE:

```
SELECT Order1.Order_Id, Tracking_Details.Courier_Name
FROM Order1
INNER JOIN Tracking_Details ON Order1.Tracking_Number = Tracking_Details.Tracking_Number;
```

OUTPUT:

Order_Id	Courier_Name
1	FastCourier
2	QuickShip
3	SpeedyDelivery
4	ExpressDelivery
5	SwiftCourier
6	RapidShip
7	ZoomDelivery
8	SpeedyShip
9	LightningCourier
10	FlashDelivery

16) Retrieve all employees with a salary greater than the average salary.

CODE:

```
SELECT * FROM Employee
WHERE Salary > (SELECT AVG(Salary) FROM Employee);
```

OUTPUT:

Emp_code	Emp_name	Date	Month	Year	Age	Position	Salary	Username
1	Alex	2024-01-01	January	2024	30	Manager	5000	alex123
5	Emma	2024-05-05	May	2024	35	Manager	5500	emma123
9	Ivy	2024-09-09	September	2024	32	Manager	5200	ivy123
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

17) Retrieve the products that have a stock of less than 50.

CODE:

```
SELECT Item_Name, Stock
FROM Products
WHERE Stock < 50;
```

OUTPUT:

Item_Name	Stock
Chicken	30

18) Retrieve the total sales amount for each product category.

CODE:

```
SELECT Product_Category.Category_Name, SUM(Bill.Total_amount) AS Total_Sales
FROM Product_Category
INNER JOIN Products ON Product_Category.Item_Code = Products.Item_Code
INNER JOIN Bill ON Products.Item_Code = Bill.Item_Code
GROUP BY Product_Category.Category_Name;
```

OUTPUT:

Category_Name	Total_Sales
Dairy	75
Bakery	15
Fruits	5
Beverages	20
Sweets	10
Grains	30
Meat	40
Vegetables	80

19) Retrieve products and their advertising company names

CODE:

```
SELECT Products.Item_Name, Advertising_Company.Company_Name
FROM Products
JOIN Advertises ON Products.Item_Code = Advertises.Item_Code
JOIN Advertising_Company ON Advertises.Company_Code = Advertising_Company.Company_Code;
```

OUTPUT:

	Item_Name	Company_Name
▶	Milk	FreshAds
	Bread	QuickAds
	Apple	FruitAds
	Orange Juice	DrinkAds
	Chocolate	SweetAds
	Rice	GrainAds
	Chicken	MeatAds
	Tomato	VeggieAds
	Potato	TaterAds
	Eggs	EggAds

20) Retrieve the most expensive product

CODE:

```
SELECT * FROM Products WHERE Item_Price = (SELECT MAX(Item_Price) FROM Products);
```

OUTPUT:

	Item_Code	Item_Name	Item_Price	GST	Discount_Offer	Stock
▶	6	Rice	10	0	0	50
*	NULL	NULL	NULL	NULL	NULL	NULL

21) Calculate the average price of products

CODE:

```
SELECT AVG(Item_Price) FROM Products;
```

OUTPUT:

	AVG(Item_Price)
▶	3.0000

22) Calculate the total stock of all products

CODE:

```
SELECT SUM(Stock) FROM Products;
```

OUTPUT:

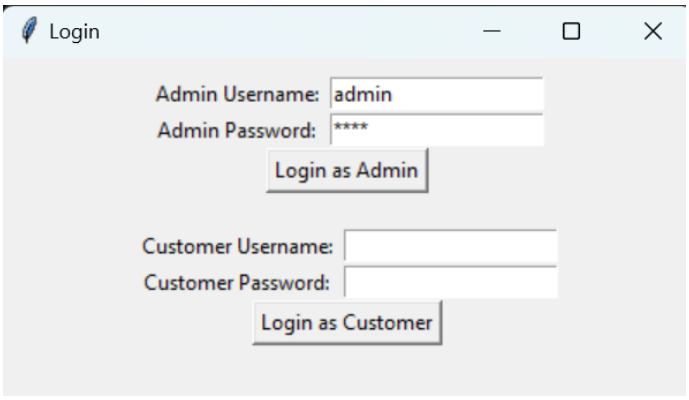
	SUM(Stock)
▶	1350

VI. Project demonstration

- GUI developed on Python
- Database developed on mySQL
- Tkinter: used to generate Bill
- Mysql.connector: used to connect the Python and Mysql to access database for GUI.
- Getpass : The getpass module provides a secure way to handle the password prompts where programs interact with the users via the terminal.

Login Interface :-

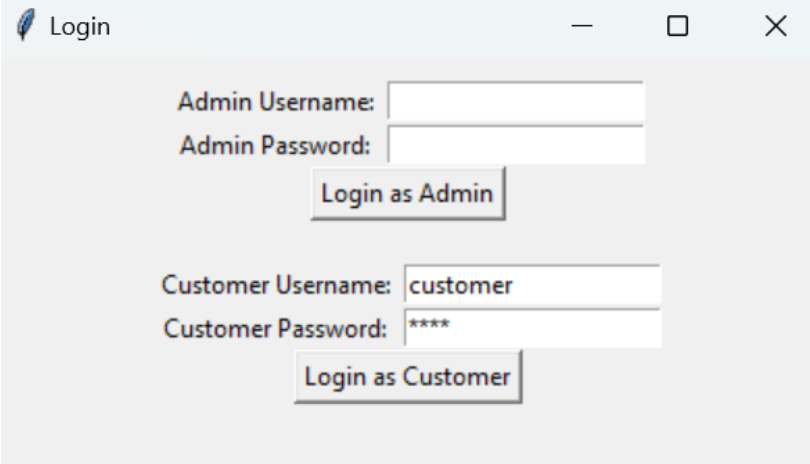
Admin –




```
Admin Menu:
1. Product Information
2. Employee Information
3. Exit
Enter Your Choice: 1

Product Information Menu:
1. Create Table
2. Insertion
3. Updation
4. Deletion
5. Delete All Products
6. Display One Product
7. Display All Products
8. Back to Admin Menu
Enter your choice: 7
Enter table name: products
+-------------+--------------+--------------+-------+------------------+------------+
|   Item_Code | Item_Name    |   Item_Price |   GST |   Discount_Offer |   Quantity |
|-------------+--------------+--------------+-------+------------------+------------|
|           1 | Milk         |            3 |     0 |                0 |        100 |
|           2 | Bread        |            2 |     0 |                0 |        150 |
|           3 | Apple        |            1 |     0 |                0 |        200 |
|           4 | Orange Juice |            3 |     0 |                0 |         80 |
|           5 | Chocolate    |            2 |     0 |                0 |        120 |
|           6 | Rice         |           10 |     0 |                0 |         50 |
|           7 | Chicken      |            8 |     0 |                0 |         30 |
|           8 | Tomato       |            0 |     0 |                0 |        220 |
|           9 | Potato       |            0 |     0 |                0 |        300 |
|          10 | Eggs         |            1 |     0 |                0 |        100 |
+-------------+--------------+--------------+-------+------------------+------------+
```

```
Admin Menu:
1. Product Information
2. Employee Information
3. Exit
Enter Your Choice: 2

Employee Information Menu:
1. Enter Employee Details
2. Display Employee Details
3. Delete Employee Details
4. Back to Admin Menu
Enter Task No. : 2
                    EmpCode      EmpName    Age  DateOfJoining      Position  Salary
--  -------  ----------  ---------  ---------  -----  ---------------  ----------  ----------
 1  Alex     2024-01-01  January         2024     30  Manager                5000  alex123
 2  Brian    2024-02-02  February        2024     25  Salesperson            3000  brian123
 3  Charlie  2024-03-03  March           2024     22  Cashier                2500  charlie123
 4  David    2024-04-04  April           2024     28  Storekeeper            3200  david123
 5  Emma     2024-05-05  May             2024     35  Manager                5500  emma123
 6  Frank    2024-06-06  June            2024     26  Salesperson            3100  frank123
 7  Grace    2024-07-07  July            2024     23  Cashier                2600  grace123
 8  Henry    2024-08-08  August          2024     29  Storekeeper            3300  henry123
 9  Ivy      2024-09-09  September       2024     32  Manager                5200  ivy123
10  Jack     2024-10-10  October         2024     27  Salesperson            2900  jack123

Employee Information Menu:
1. Enter Employee Details
2. Display Employee Details
3. Delete Employee Details
4. Back to Admin Menu
Enter Task No. :
```

Customer Login -



```
Customer Menu:
1. View Products
2. Exit
Enter Your Choice: 1
+-------------+--------------+--------------+-------+------------------+------------+
|   Item_Code | Item_Name    |   Item_Price |   GST |   Discount_Offer |   Quantity |
|-------------+--------------+--------------+-------+------------------+------------|
|           1 | Milk         |            3 |     0 |                0 |        100 |
|           2 | Bread        |            2 |     0 |                0 |        150 |
|           3 | Apple        |            1 |     0 |                0 |        200 |
|           4 | Orange Juice |            3 |     0 |                0 |         80 |
|           5 | Chocolate    |            2 |     0 |                0 |        120 |
|           6 | Rice         |           10 |     0 |                0 |         50 |
|           7 | Chicken      |            8 |     0 |                0 |         30 |
|           8 | Tomato       |            0 |     0 |                0 |        220 |
|           9 | Potato       |            0 |     0 |                0 |        300 |
|          10 | Eggs         |            1 |     0 |                0 |        100 |
+-------------+--------------+--------------+-------+------------------+------------+

Customer Menu:
1. View Products
2. Exit
Enter Your Choice: 2
Exiting...
```

Bill :-

VII. Self -Learning beyond classroom

:

1. **Normalization Techniques**: Understanding normalization techniques like 1NF, 2NF, and 3NF and applying them to ensure the database is structured efficiently and without redundancy.
2. **Transaction Management**: Learning about ACID (Atomicity, Consistency, Isolation, Durability) properties and how to manage transactions to ensure data integrity and consistency.
3. **Database Indexing**: Learning about indexing techniques to optimize the performance of database queries, especially for large datasets.
4. **Backup and Recovery**: Understanding backup and recovery strategies to safeguard the data and restore it in case of any failures.
5. **Data Validation and Integrity Constraints**: Implementing data validation checks and integrity constraints to ensure the accuracy and reliability of the data.
6. **SQL Query Optimization**: Learning techniques to optimize SQL queries to improve the performance of database operations.

VIII. Learning from the Project

The Grocery Store Management System project has been an invaluable learning experience, offering numerous insights and enhancing my skills in several key areas:

1. Database Design and Management: The project provided hands-on experience in designing, implementing, and managing a complex relational database system. It allowed me to understand the importance of database normalization, indexing, and optimization techniques to ensure efficient data storage and retrieval.

2. SQL Query Optimization: Working on this project required writing and optimizing various SQL queries to perform data retrieval, insertion, and modification operations. This experience was instrumental in improving my SQL skills and understanding the significance of query optimization for enhancing database performance.

3. Understanding Business Requirements : The project enabled me to understand and analyze the business requirements of a grocery store and translate them into a functional database design. It taught me the importance of aligning the database structure with the operational needs of the business to ensure smooth and efficient operations.

4. Data Integrity and Validation : Implementing data validation checks and integrity constraints in the database was a crucial aspect of the project. It taught me the importance of maintaining data accuracy and consistency by enforcing validation rules and integrity constraints.

5. Systematic Approach to Problem-Solving : The project required a systematic approach to problem-solving, from identifying the entities, attributes, and relationships to implementing the database design and resolving any issues or

errors that arose during the development process. This experience honed my problem-solving skills and taught me to approach complex problems in a structured and organized manner.

6. Project Management Skills: Managing the project involved planning, organizing, and executing various tasks, including database design, implementation, testing, and documentation. This experience enhanced my project management skills and taught me the importance of effective project planning and coordination to ensure the successful completion of the project within the specified timeframe.

7. Learning Beyond the Classroom: The project encouraged me to explore and learn new concepts and techniques beyond the classroom, such as transaction management, backup and recovery strategies, database security, and user interface design. This self-directed learning enhanced my overall understanding of database management and provided me with valuable practical skills that are applicable in real-world scenarios.

8. Teamwork and Collaboration: Although the project was an individual effort, it taught me the importance of teamwork and collaboration in a larger organizational context. It helped me appreciate the significance of clear communication, cooperation, and coordination with stakeholders, including potential end-users, to ensure the successful implementation and adoption of the Grocery Store Management System.

In conclusion, the Grocery Store Management System project has been a rewarding and enriching experience that has significantly contributed to my academic and professional development. It has equipped me with valuable technical skills, practical experience, and a deeper understanding of database management, preparing me to tackle more complex and challenging projects in the future.

IX. Challenges Faced

Challenges Faced

1. **Database Design Complexity**:
    - Designing a comprehensive and efficient database system for a grocery store involved dealing with complex relationships among various entities, which posed challenges in ensuring proper normalization, data integrity, and efficient query processing.
2. **Data Consistency and Integrity**:
    - Ensuring consistent and accurate data entry and maintenance posed challenges in implementing effective validation checks and integrity constraints to prevent data redundancy and inconsistencies.
3. **Optimizing Query Performance**:
    - Optimizing the performance of SQL queries to handle large datasets and complex joins presented challenges in identifying and implementing efficient indexing and query optimization techniques.
4. **System Integration and Compatibility**:
    - Integrating the database management system with other systems and ensuring compatibility and seamless data flow posed challenges in coordinating and synchronizing the data across different platforms and applications.
5. **User Interface Design and User Experience**:
    - Designing an intuitive and user-friendly interface for the database management system to enhance user experience posed challenges in understanding and implementing effective UI/UX design principles and functionalities.
6. **Time Management and Deadline Pressure**:
    - Managing time effectively and meeting project deadlines posed challenges in prioritizing tasks, allocating resources, and ensuring timely completion of the project without compromising the quality and functionality of the database management system.

X. Conclusion

The Grocery Store Management System project has been an invaluable experience, offering a deep dive into the practical application of database management concepts. This endeavor not only solidified theoretical knowledge acquired in the classroom but also provided an opportunity to tackle real-world challenges in system design, data integrity, and performance optimization.

Through this project, there was a significant enhancement in problem-solving abilities, project management skills, and a better understanding of business requirements. The complexities encountered during the design phase, such as managing intricate database relationships, ensuring data consistency and integrity, and optimizing query performance, were instrumental in honing analytical and technical skills.

Despite the challenges faced, including the intricacy of database design, the intricacies of ensuring data consistency, and the demand for efficient query performance, the project was a comprehensive success. It underscored the importance of effective time management, teamwork, and the iterative nature of system development.

In conclusion, the Grocery Store Management System project has been a pivotal learning experience, bridging the gap between theoretical knowledge and practical application. It has provided a solid foundation in database management and system development, equipping with the skills and confidence to undertake more complex and challenging projects in the future.