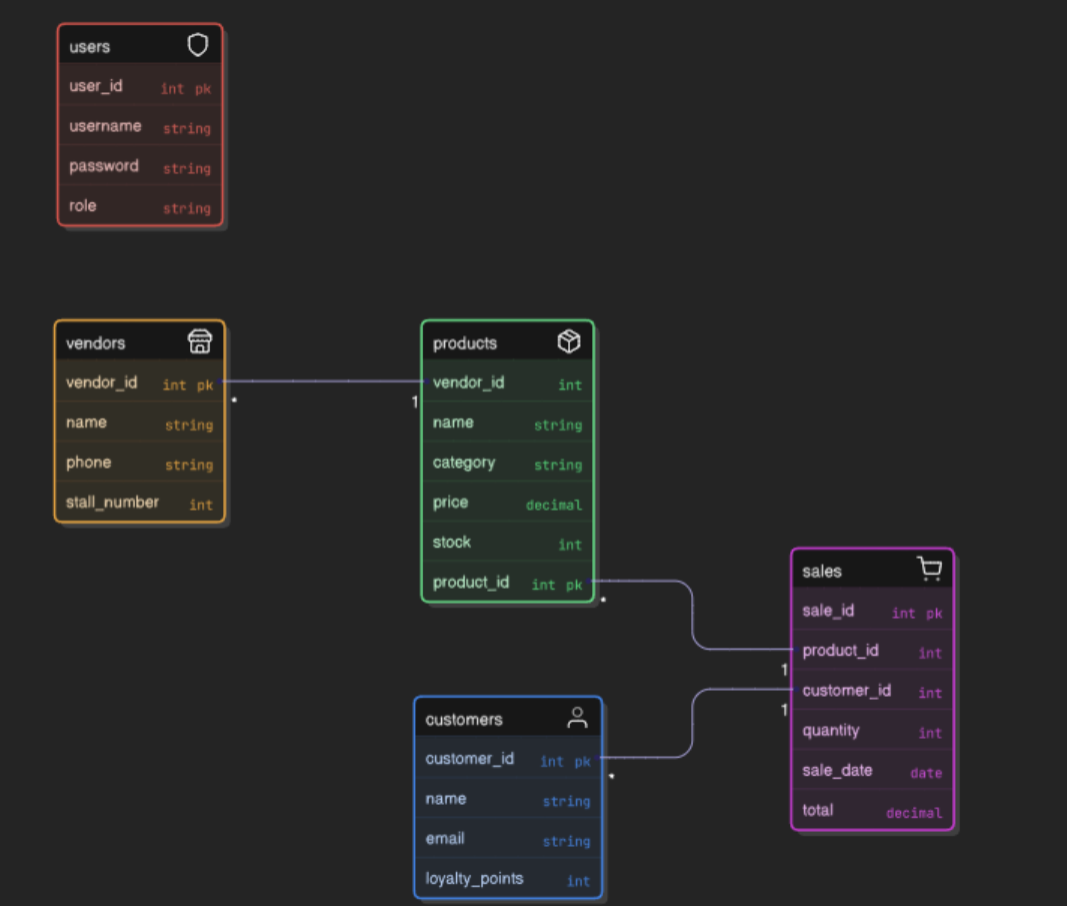
#### Final Project Report: Farmers Market Management System

##### **Introduction**

The Farmers Market Management System is a database application to manage vendors, products, customers, sales, and user access for a farmers market. It solves inefficiencies in manual record-keeping by automating inventory, sales, and customer loyalty tracking. Built with MySQL and Tkinter (Python), it includes CRUD operations, authentication, search, reporting, and data export.

##### **Database Design**

* **Schema**:
  + Vendors: vendor\_id (PK), name, phone, stall\_number (UNIQUE).
  + Products: product\_id (PK), vendor\_id (FK), name, category, price, stock.
  + Customers: customer\_id (PK), name, email (UNIQUE), loyalty\_points.
  + Sales: sale\_id (PK), product\_id (FK), customer\_id (FK), quantity, sale\_date, total.
  + Users: user\_id (PK), username (UNIQUE), password, role (Admin/Staff).
* **Relationships**: Vendors 1:N Products, Products 1:N Sales, Customers 1:N Sales.
* **Normalization**: Achieved BCNF (1NF: atomic attributes; 2NF: no partial dependencies; 3NF: no transitive dependencies).
* **ER Diagram**:



##### **Implementation**

* **Tech Stack**: MySQL, Tkinter (Python), mysql-connector-python, bcrypt, csv.
* **UI Features**:
  + **Login**: Authenticate users (Admin/Staff roles).
  + **Tabs**: Products, Vendors, Customers, Sales, Users (Admin-only), Reports.
  + **CRUD**: Full operations for all entities.
  + **Search**: By name (products, vendors, customers), by date (sales).
  + **Reports**: Total sales, sales by vendor, loyalty summary.
  + **Export**: Sales to CSV.
* **Key Code**: def add\_sale(self):  
   # Validate inputs, update stock, add sale, increment loyalty points  
   self.cursor.execute("INSERT INTO Sales ...")  
   self.cursor.execute("UPDATE Products SET stock=...")  
   self.cursor.execute("UPDATE Customers SET loyalty\_points=...")  
   self.db.commit()

##### **Additional Features**

* **Authentication**: Admin (full access), Staff (view-only).
* **Search/Reporting**: Search across entities; reports for sales and loyalty.
* **Stock & Loyalty**: Sales update stock and award points (1 per $10 spent).
* **Export**: Sales data to CSV.

##### **Challenges & Solutions**

* **Challenge**: Stock/loyalty updates during sales.

**Solution**: Used atomic transactions (db.commit()).

* **Challenge**: Role-based access.

**Solution**: Restricted actions based on current\_user\_role.

* **Challenge**: Input validation.

**Solution**: Added try-except for numeric/date inputs.

##### **Conclusion**

The system meets all requirements, offering a practical solution for farmers market management with a normalized database, secure authentication, and robust features. Tested with sample data, it handles all operations and edge cases effectively.