CS5200 Database Management System
Final Project Report

Team name:

ZhuYChengC

Team members:

Yanrun Zhu, Chia-Sheng Cheng

Final Project Demo Link:

https://www.youtube.com/watch?v=2P1l36voKMM

README

Project Setup and Installation:
Programming Language: Python 3.x
- Download Python 3.x from Python's official website.
- Ensure pip (Python's package manager) is installed with Python.

Database: MySQL
- Install MySQL Server from MySQL's official download page.
- Set up your MySQL environment (username, password, and database).

Dependencies:
- PyMySQL: Run pip install pymysql to install the PyMySQL library for Python.

Installation Directories:
- Clone or download the project to a directory of your choice.
- Example: C:\Users\YourName\BeverageStoreManagementSystem or /home/yourname/BeverageStoreManagementSystem

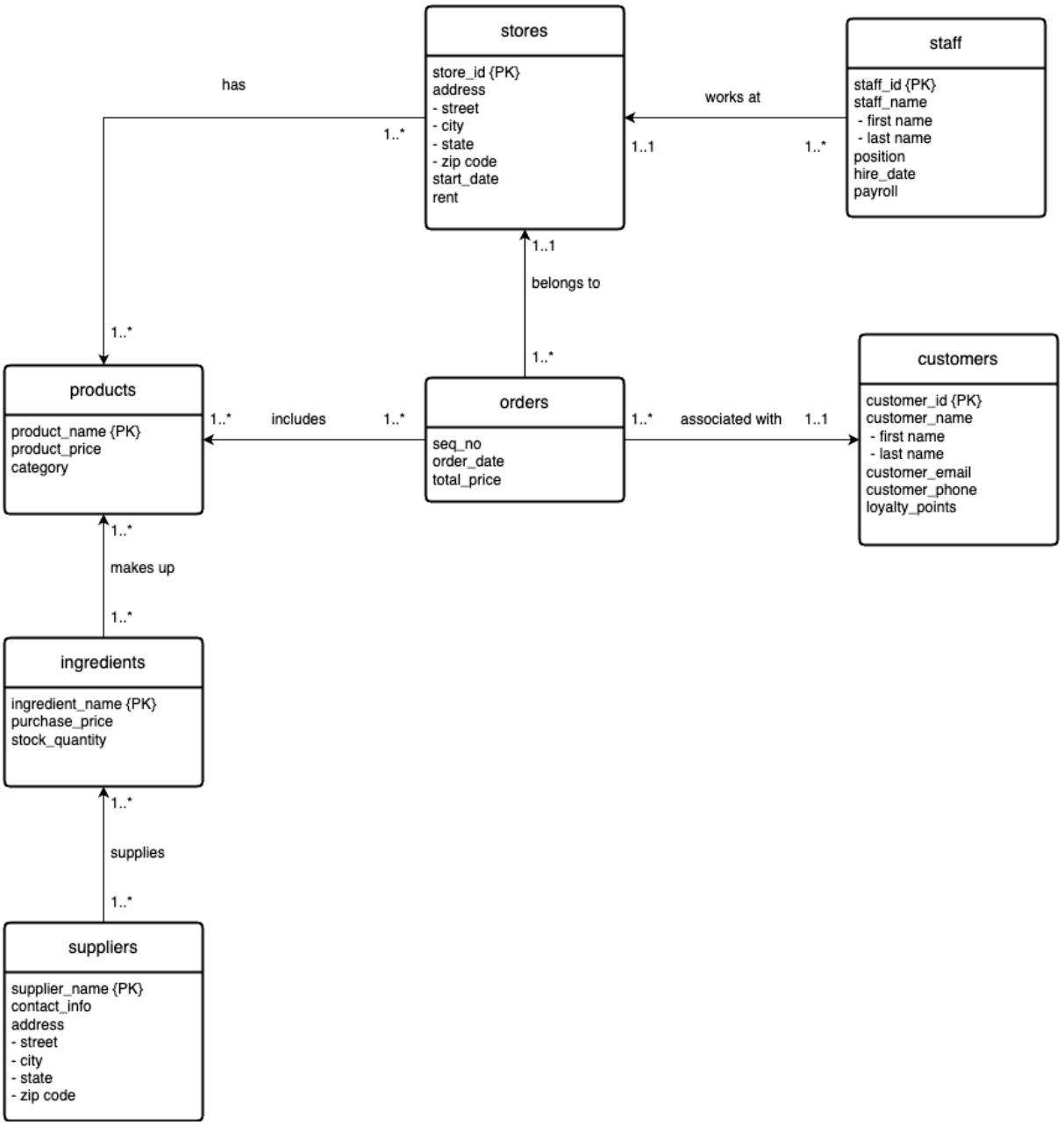
Running the Application:
- Navigate to the project directory in your command line or terminal.
- Run python main.py to start the application.

Technical Specifications

Language: Python 3.x
Framework: No specific framework used. Core Python libraries and PyMySQL for database interaction.
Database: MySQL (Relational Database Management System)
File Structure:

- main.py: The main script that users interact with. It initiates the application and handles the primary user interface for managing the beverage store.
- functions.py: This script contains the BeverageStoreApp class and related functions. It includes methods for connecting to the MySQL database, executing SQL queries, and performing various operations related to store management.
- beverage_store_dump.sql: This SQL file is used for initializing the database schema and possibly contains initial data for the beverage store. It includes SQL statements for creating tables, inserting data, and setting up the database environment.

Conceptual Design (UML)



Data Description:

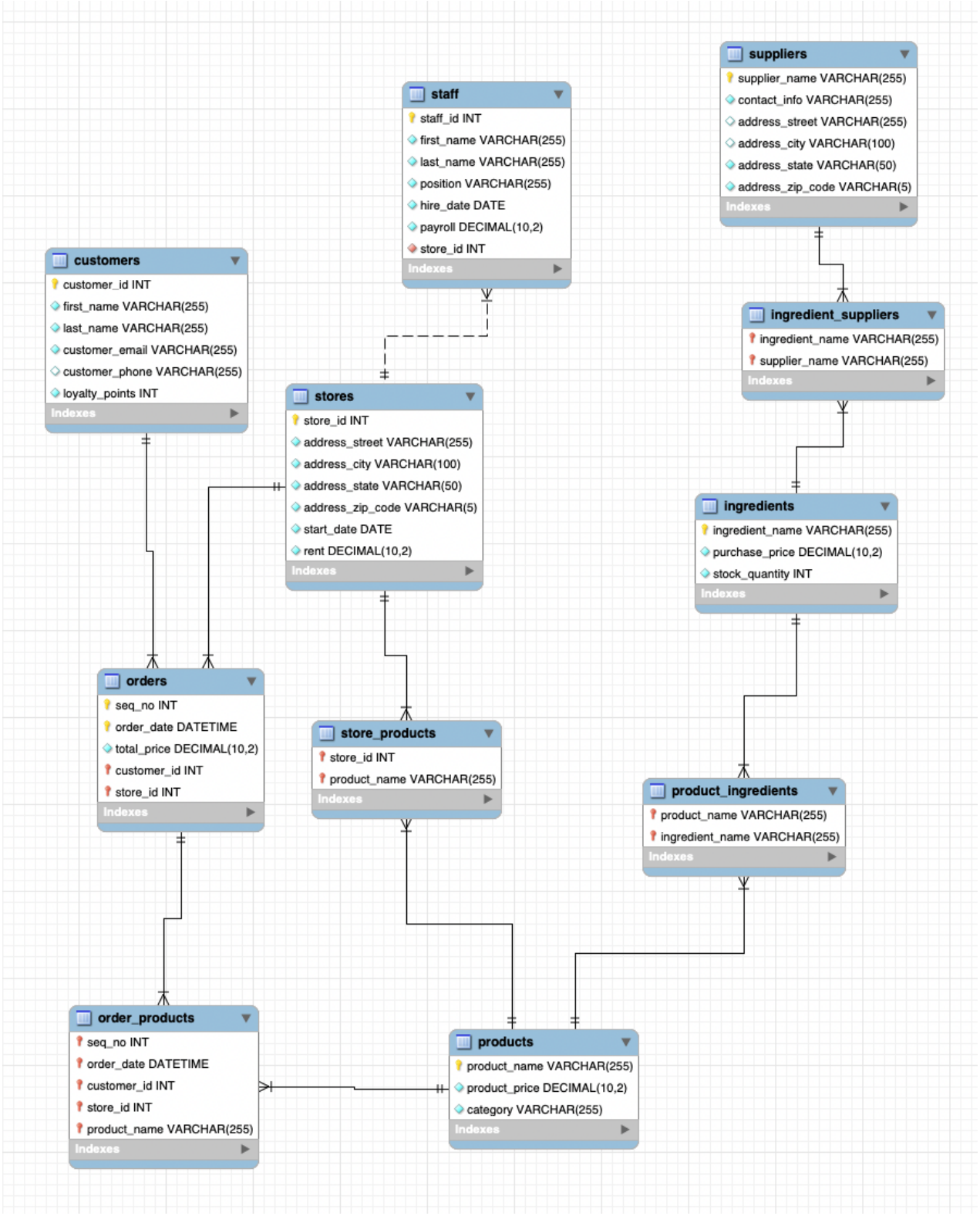
- Stores: This entity contains details about each store, including a unique store ID, address, start date, and the rent for the store.
- Staff: This represents the employees working at each store, with a staff ID, name, position, hire date, payroll, and a reference to the store they work at.
- Customers: It stores customer information, such as a customer ID, name, email, phone number, and loyalty points.
- Orders: This entity tracks each order placed, identified by a sequence number, order date, total price, and references to the customer who placed the order and the store where the order was placed.
- Products: It catalogs the products sold, with a product name, price, and category.
- Ingredients: This details the ingredients used in the products, with an ingredient name, purchase price, and stock quantity.
- Suppliers: It lists the suppliers providing the ingredients, with a supplier name, contact information, and address.

Relationships Description:

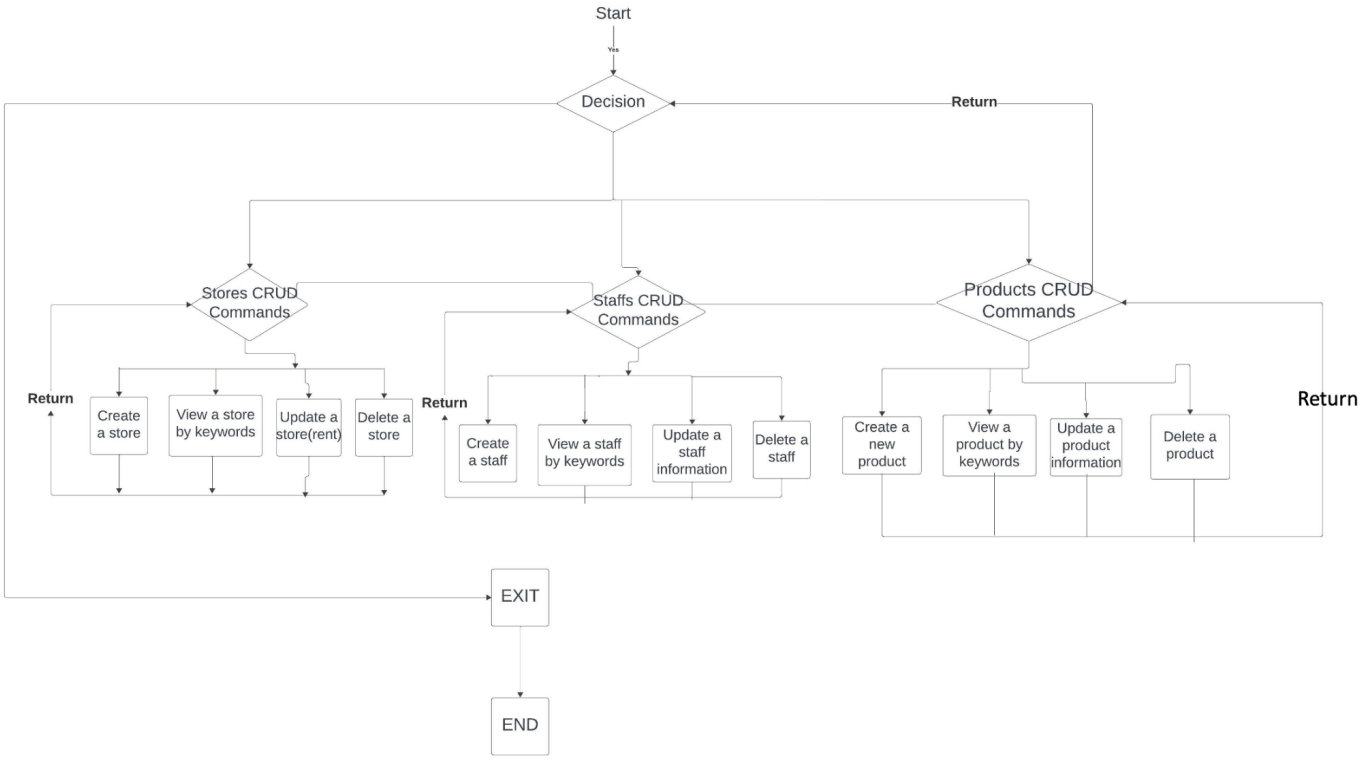
- Stores to Staff: A one-to-many relationship. Each store may have multiple staff members, but each staff member works at only one store. This relationship is managed by a foreign key in the staff table that references the store_id in the stores table.
- Stores to Orders: A one-to-many relationship. Each store can have multiple orders placed from it, but each order is associated with one store. This is implemented by a foreign key in the orders table that references the store_id from the stores table.
- Customers to Orders: A one-to-many relationship. A customer can place multiple orders, but each order is linked to a single customer. The orders table has a foreign key that references the customer_id from the customers table.
- Products to Orders: A many-to-many relationship. An order can contain multiple products, and a product can be part of multiple orders. The order_products table serves as a junction table with foreign keys referencing seq_no, order_date, customer_id, and store_id from the orders table, and product_name from the products table.
- Products to Ingredients: A many-to-many relationship. A product can be made up of multiple ingredients, and an ingredient can be used in multiple products. The product_ingredients table acts as a junction table containing foreign keys that reference the product_name from the products table and the ingredient_name from the ingredients table.
- Stores to Products: A many-to-many relationship. A store can carry many different products, and a product can be stocked by multiple stores. The store_products table is the junction table with foreign keys that reference the store_id from the stores table and the product_name from the products table.
- Ingredients to Suppliers: A many-to-many relationship. A supplier can supply multiple ingredients, and an ingredient can be provided by multiple suppliers.

This is managed by the ingredient_suppliers junction table, which has foreign keys referencing ingredient_name from the ingredients table and supplier_name from the suppliers table.

Logical Design:



User Flow:



Here's a step-by-step rundown of the commands or methods a user would perform to interact with the system:

1. Start the Application
    - User runs main.py.
2. Main Menu
    - The user is prompted to enter their MySQL username.
    - The application displays the main menu with options to manage:
        1) Store
        2) Staff
        3) Product
        4) Exit
    - The user enters a choice (1-4).
3. Store Management
    - If 1 is chosen, the store_menu function is invoked:
        1) Create Store: app.create_store()
        2) View Stores: Leads to view_stores_menu
        3) Update Store: app.update_store_rent()
        4) Delete Store: app.delete_store()
        5) Return to Main Menu
    - In view_stores_menu, the user can view stores by ID, state, city, or detailed info.
4. Staff Management
    - If 2 is chosen, the staff_menu function is invoked:
        1) Create Staff: app.create_staff()

2) View Staff: Leads to view_staff_menu
3) Update Staff: Leads to update_staff_menu
4) Delete Staff: app.delete_staff()
5) Return to Main Menu

- In view_staff_menu, the user can view staff by position or store.
- In update_staff_menu, the user can update staff's position, payroll, or workplace.

5. Product Management
   - If 3 is chosen, the product_menu function is invoked:
     1) Create Product: app.create_product()
     2) View Products: Leads to view_products_menu
     3) Update Product: app.update_product_price()
     4) Delete Product: app.delete_product()
     5) Return to Main Menu
   - In view_products_menu, the user can view products by category, price range, or ingredient.
6. Exit Application
   - If 4 is chosen at the main menu, the application prints "Exiting the application." and the main loop breaks.
7. Close Database Connection
   - Once the user decides to exit, the application closes the database connection.

Throughout the user interaction, if an invalid choice is entered, the application will print "Invalid choice. Please try again." and display the menu again for the user to make a correct selection. This user flow allows for a comprehensive management of the beverage store's operational data through a command-line interface.

Lessons Learned :

1. Technical Expertise Gained:

Throughout the course of this project, we have developed a robust set of technical skills in both Python and MySQL, enhancing our capabilities as a data scientist. The key technical expertise gained includes:

- Seamless integration of Python scripts with MySQL database for data extraction and loading.
- Effective use of MySQL Connector to establish and manage database connections within Python.
- Utilization of version control systems (e.g., Git) for efficient collaboration and tracking of project changes.
- Experience with collaborative coding through platforms like GitHub, facilitating team-based development. scripts.

2. Insights, Time Management, and Data Domain Insights:

- Acquired a deep understanding of the project's domain, including its unique challenges and opportunities.

- Identified key patterns and trends within the data, leading to valuable insights for decision-making.
- Developed effective time management skills by adhering to project timelines and milestones.
- Prioritized tasks based on project requirements and adjusted schedules to accommodate unforeseen challenges.
- Gained domain-specific knowledge, allowing for informed data preprocessing decisions.
- Recognized the significance of context in data interpretation, leading to more meaningful analyses.
- Improved communication skills through regular project updates and effective reporting to stakeholders.
- Developed the ability to convey complex technical concepts to non-technical audiences.

3. Realized or Contemplated Alternative Design/Approaches:
    - Considered alternative database systems (e.g., MongoDB) based on the nature of the data and project requirements.
    - Evaluated the trade-offs between relational and NoSQL databases and their implications for scalability.
    - Explored alternative machine learning algorithms and models to compare predictive performance.
    - Considered ensemble methods and deep learning architectures for potential improvements in accuracy.
    - Evaluated the scalability of the chosen database and explored the use of distributed computing frameworks for large-scale data processing.
    - Investigated strategies for optimizing query performance as the dataset grows.
4. Document any code not working in this section:
    - In our project, all codes work successfully.

In conclusion, this project has not only enhanced our technical proficiency in Python and MySQL but has also provided valuable insights into effective time management, domain-specific considerations, and alternative approaches to solving data science challenges. These experiences have contributed significantly to our growth as a data science professionals, preparing us for future projects and challenges in the field.

Future work

1. Planned uses of the database:
    - Advanced Analytics and Reporting: The database could be utilized for generating detailed analytics on sales trends, staff performance, and inventory management. This would help in making informed business decisions.

- Integration with Point-of-Sale (POS) Systems: The current database can be expanded to integrate with POS systems across different store locations to facilitate real-time data synchronization and streamline sales processes.
- Customer Relationship Management (CRM): The database can be developed further to support CRM functionalities, tracking customer interactions, and preferences to enhance customer service and targeted marketing.
- Supply Chain Optimization: By analyzing the inventory and supplier data, the database could be used to optimize the supply chain, reduce costs, and improve efficiency.
- E-commerce Platform Support: As the retail landscape evolves, the database could serve as the backend for an e-commerce platform, allowing the business to sell products online.

2. Potential areas for added functionality:
   - User Authentication: Implementing user authentication to provide different access levels to the application, enhancing security, and ensuring data integrity.
   - Mobile Application Interface: Developing a mobile application interface that connects to the database for on-the-go management and access.
   - Automated Reordering System: A functionality that automatically places orders for inventory based on predefined thresholds could minimize stockouts and overstocking.
   - Feedback and Review System: Incorporating a feedback mechanism where customers can rate products and services, which can then be stored in the database for quality improvement and analysis.
   - Employee Scheduling: Adding a module for staff management, including scheduling shifts, tracking work hours, and managing payroll within the database system.
   - Loyalty Program Management: Enhancing the customer module to manage loyalty programs more effectively, including tracking points and issuing rewards.
   - Business Intelligence (BI) Integration: The database can be structured to support BI tools for complex data visualizations and to uncover deeper insights into business operations.

These planned uses and potential functionalities aim to not only expand the capabilities of the Beverage Store Management System but also to ensure it can adapt to future business needs and technological advancements.