A Supermarket Management Database System Based on Entity-Relationship Model

Yifei DAI 121090093
Kaiying HAN 121090155
You LV 121090404
Tengfei MA 121090406
Tieding MA 121090407
Sixuan MAO 121090414
Ziyu XIE 121090642

1. Introduction and Motivation

In response to the challenges of managing vast data volumes that supermarkets generate daily, we developed a comprehensive supermarket management database system. Traditional data management methods often led to inefficiencies, such as delayed information retrieval and data inconsistencies. Our system integrates key subsystems—Product Management, Product Supply, Product Sales, Employee Management, and Marketing Reporting—into a unified framework, enhancing data flow and accessibility. It ensures data integrity, independence, and efficient access, minimizing redundancy and effectively scaling with growth. Additionally, the system incorporates large language models (LLMs) to refine functionality, facilitating intelligent data handling and advanced query generation for precise data mining. This technologically advanced system not only improves the efficiency but also the effectiveness of supermarket management, demonstrating the transformative potential of integrated database systems in modern retail.

In our report, we first analyze the requirements of the organization, identify the relevant entities, attributes, and relationships together with any constraints and properties, produce an E-R diagram for the database and use LLM to refine our design (shown as subtitle **2.1, 2.2**). Followed by generating and evaluating our schema (subtitle **2.3**). We also considered indexing and hashing strategies of our data field (subtitle **2.4**) and how we generate our data (subtitle **2.5**). Then we introduced our implementations of sample SQL queries that are used for practical daily operations and activities (subtitle **3.1**) and of analytic or data mining nature (subtitle **3.2**). Finally, we investigate methods for crafting prompts that guide LLM to accurately generate queries for extracting information from the database (title **4**).

2. Database Design

2.1 LLM Helps Refining Database Architecture

The initial design (Fig.1) of our supermarket management database system was conceptualized as a complex network of interrelated tables, capturing various aspects of the supermarket operations such as inventory, sales, supplies, employee details, and customer interactions. This initial diagram served as a broad canvas, illustrating all potential data points and relationships that we considered important for an effective management system.

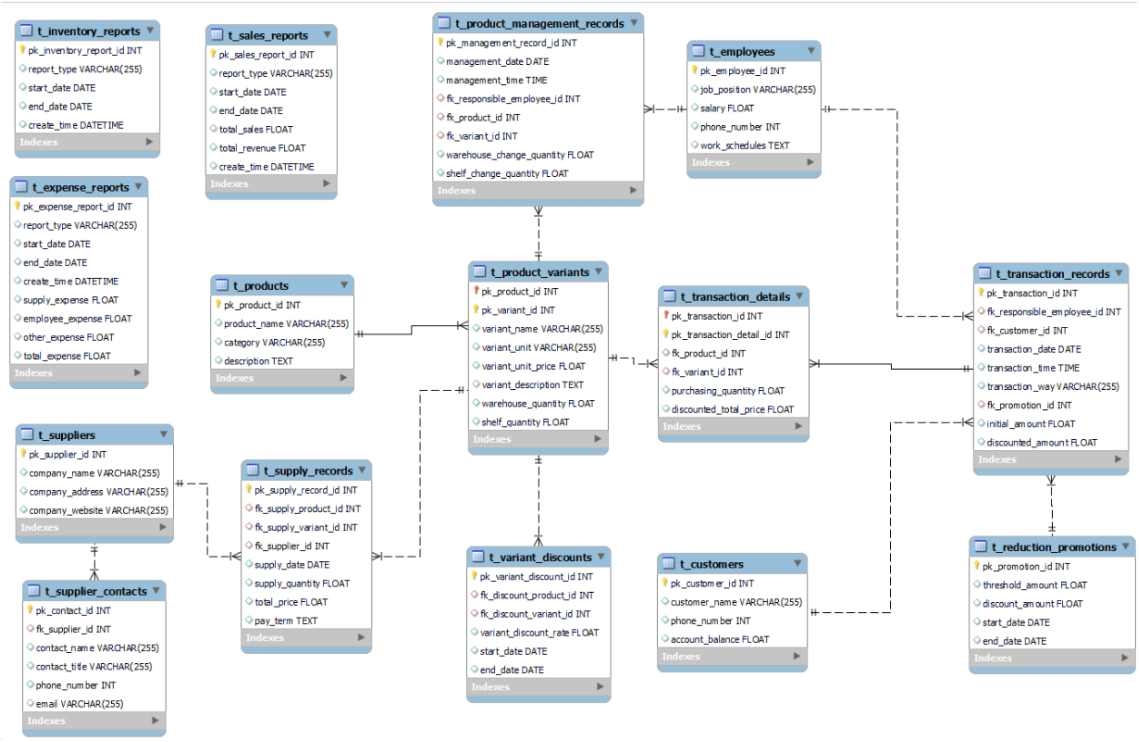

Figure 1: Initial ER diagram

As the design process progressed, the integration of a large language model (LLM) played a pivotal role in refining our database structure. Leveraging the analytical capabilities of the LLM, we critically evaluated the logical relationships between tables. This analysis allowed us to identify and eliminate redundancies—such as duplicated data fields and unnecessary relational links—that could complicate queries and data maintenance. For instance, the LLM helped pinpoint overlapping attributes across tables that were consolidated to streamline the data model.

Moreover, the LLM's insights facilitated a more structured approach to our database design by assisting in the categorization of tables into distinct modules based on their functional relevance. Each module—Product Management, Product Supply, Product Sales and Employee Management—was designed to encapsulate a specific subset of the supermarket's operations. This modularization not only enhanced the clarity of the database schema but also improved maintainability and scalability. By focusing on module-specific

interactions, we could tailor optimizations and security measures appropriate for each segment, enhancing overall system performance.

The final ER diagram (Fig.2), as a result, represents a more streamlined and efficient database system. It is a refined blueprint that clearly delineates the relationships and dependencies between tables, structured in a way that optimizes data retrieval and manipulation. This final design is not just a reflection of technical improvements but also a testament to strategic planning, facilitated by advanced computational tools like the LLM, ensuring that the supermarket management system is both robust and adaptable to future needs.

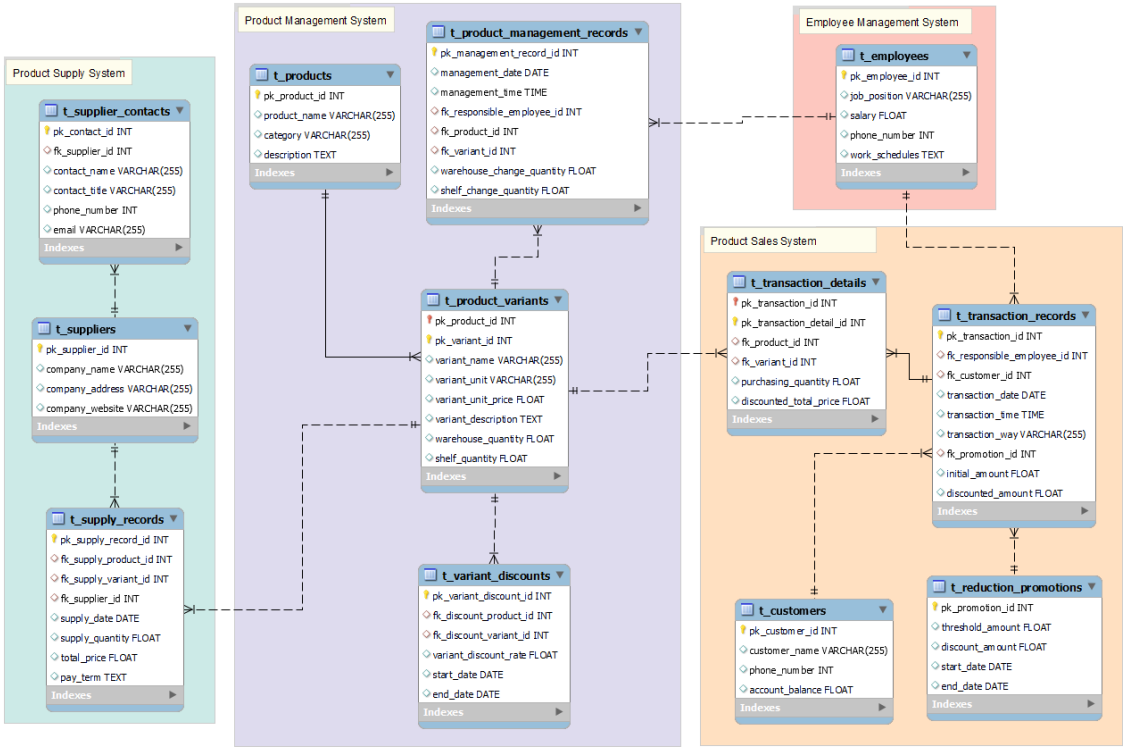

Figure 2: Final ER diagram refined by LLM

2.2 Detailed Information About the ER Diagram

As we mentioned above, four modules are designed to encapsulate a specific subset of the supermarket's operations. The introduction of each module is shown below.

2.2.1 Product Management System

The Product Management System depicted in the provided ER diagram is a crucial component of the supermarket management database. It is designed to handle various aspects of product inventory, ranging from the management of product information to inventory control and discount offerings. Below is a detailed explanation of the function of the system, its relationship with other systems, and the role of each table within this subsystem.

The primary function of the Product Management System is to manage detailed information about the products offered in the supermarket. This includes tracking product variants, managing inventory levels both in the warehouse and on store shelves, and overseeing discount strategies to optimize sales and inventory turnover. The system ensures that product information is up-to-date, accurately reflecting current stock levels, and that pricing and discount strategies are effectively implemented. The Product Management System, a critical subsystem within the supermarket management database, encompasses several tables designed to handle all aspects of product information, inventory management, and discount strategies, ensuring comprehensive management and real-time accuracy in product-related data.

The **t_products** table acts as the core repository for product data. The attribute *pk_product_id* uniquely identifies each product variant, essential for distinguishing different sizes or flavors of the same product. The *product_name* records the common name of the product as recognized in the supermarket. *category* classifies products into various supermarket sections like dairy or beverages, facilitating organized shelf layout and inventory searches. The *description* provides brief details about the product, such as taste, use, or any special attributes appealing to consumers.

Adjacent to this is the **t_product_variants** table, which manages the variations of each product. Here, *pk_variant_id* serves as a primary key, again ensuring each product variant is uniquely cataloged. *variant_name*, *variant_unit*, and *variant_unit_price* detail the specific characteristics of each variant, including its name, measurement unit, and pricing respectively. *variant_description* offers more detailed insights into the variant, such as flavor or additional features. The attributes *warehouse_quantity* and *shelf_quantity* are crucial for tracking the stock levels in the warehouse and on the store shelves, respectively.

Furthermore, the **t_product_management_records** table records inventory changes and is vital for inventory tracking and management. The *pk_management_record_id* uniquely identifies each record. The *management_date* and *management_time* capture when the inventory changes occurred, offering temporal precision. *fk_responsible_employee_id* links the record to an employee who managed the change, ensuring accountability. *warehouse_change_quantity* and *shelf_change_quantity* reflect the quantity adjustments in the warehouse and on the shelves, critical for accurate stock management.

Lastly, the **t_variant_discounts** table handles the discounting aspects of product variants. *pk_variant_discount_id* is the primary identifier for each discount record. It references the product and variant via *fk_discount_product_id* and *fk_discount_variant_id*. The *variant_discount_rate* specifies the discount percentage, whereas *start_date* and *end_date* define the duration of the discount, crucial for promotional activities.

Each of these tables is interconnected, supporting a robust product management sys-

tem that ensures products are well-cataloged, stock levels are meticulously managed, and pricing strategies are effectively executed to optimize sales and inventory turnover in the supermarket.

2.2.2 Product Supply System

The Product Supply System is another essential subsystem within the supermarket management database, tasked with managing the information regarding suppliers, their contacts, and supply transactions to ensure timely and efficient inventory replenishment.

The **t_suppliers** table stores crucial data about the suppliers. The attribute *pk_supplier_id* is the primary key, uniquely identifying each supplier. *company_name* provides the name of the supplier company, *company_address* details the physical address, and *company_website* offers a direct link to the supplier's website, facilitating easy access to further company details and product catalogs.

Adjacent to this, the **t_supplier_contacts** table records contact details for each supplier, crucial for communication and order management. *pk_contact_id* acts as the primary key for this table. Each contact is detailed through *contact_name* and *contact_title*, providing the name and job title of the contact person respectively. Contact details are completed by *phone_number* and *email*, ensuring multiple channels for communication.

Furthermore, the **t_supply_records** table captures each supply transaction, providing a comprehensive log of inventory movements related to supplier interactions. The primary key, *pk_supply_record_id*, uniquely identifies each transaction. This table links to suppliers and products through *fk_supplier_id*, *fk_supply_product_id*, and *fk_supply_variant_id*, ensuring traceability from the product variants to the original suppliers. Attributes like *supply_date*, *supply_quantity*, and *total_price* record the date of the transaction, the quantity of goods supplied, and the total transaction cost, respectively.

Each of these tables is intricately linked, creating a robust Product Supply System that ensures seamless supplier management, effective communication, and precise inventory replenishment, critical for maintaining optimal stock levels and ensuring the supermarket's operational efficiency.

2.2.3 Product Sales System

The Product Sales System is a critical subsystem within the supermarket management database, designed to comprehensively manage customer transactions, including sales data, promotions, and customer interactions, ensuring effective sales tracking and customer service.

The **t_transaction_records** table serves as the main repository for transaction logs. Each record is uniquely identified by *pk_transaction_id*, the primary key. The table includes references to the responsible employee and the customer through

fk_responsible_employee_id and *fk_customer_id*, respectively, linking to their respective records. Transaction specifics such as *transaction_date*, *transaction_time*, and *transaction_way* (e.g., cash, credit) are meticulously recorded to provide a complete view of the transaction context. Promotional discounts applied during the transaction are tracked via *fk_promotion_id*, with fields *initial_amount* and *discounted_amount* detailing the financial aspects before and after discounts.

Adjacent to this is the **t_customers** table, which contains detailed information specific to each customer, enhancing personalization and customer service. The primary key, *pk_customer_id*, uniquely identifies each customer. Attributes like *customer_name*, *phone_number*, and *account_balance* help in managing customer relationships and offering targeted promotions or services based on purchase history and account status.

Furthermore, the **t_transaction_details** table captures the specifics of each item within a transaction, linked by *pk_transaction_id*. It includes *pk_transaction_detail_id* as the primary key, with links to products and their variants via *fk_product_id* and *fk_variant_id*, respectively. The attribute *purchasing_quantity* records the number of units purchased, crucial for inventory and sales analysis.

Lastly, the **t_reduction_promotions** table manages the details of promotions applied to transactions. The primary key *pk_promotion_id* uniquely identifies each promotion. This table outlines the terms of discounts through attributes like *threshold_amount*, the minimum purchase necessary to trigger the discount, and *discount_amount*, the value of the discount offered. The promotion's validity is defined by *start_date* and *end_date*, essential for promotional planning and execution.

Each of these tables is intricately connected, supporting a robust Product Sales System that ensures transactions are accurately recorded, promotions are effectively managed, and customer relationships are meticulously maintained, contributing to optimal operational efficiency and customer satisfaction in the supermarket.

2.2.4 Employee Management System

The Employee Management System is a fundamental subsystem within the supermarket management database, dedicated to managing all aspects of employee information and work schedules, ensuring efficient human resource management and operational staffing.

The **t_employees** table is central to this system, serving as the repository for comprehensive employee records. The attribute *pk_employee_id* is the primary key, uniquely identifying each employee within the supermarket. *job_position* describes the role or title held by the employee, which is crucial for defining responsibilities and aligning work tasks. *salary* records the compensation details for each employee, vital for payroll processing and financial management. Contact details are captured under *phone_number*, ensuring easy communication for administrative or emergency purposes. The *work_schedules* attribute is particularly important, as it details the shifts or hours that employees are scheduled to work, supporting the planning of manpower allocation and operational coverage.

This table not only facilitates efficient human resource management but also integrates with other systems to ensure that employee roles align with the operational needs and customer service standards of the supermarket.

2.3 Relation Schema Evaluation

The relation schemas has actually been shown in our previous ER diagram. It has to be metioned that all the tables shown in our ER diagram are at least in Third Normal Form. For example, the table **t_transaction_details**(primary key in composition form) are shown below. All the other tables are like the form of this table, hence omitted here.

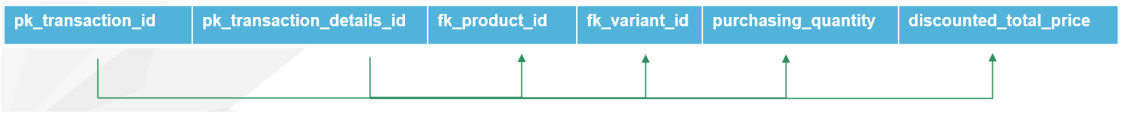

Figure 3: Table t_transaction_details

2.4 Indexing and Hashing Strategies

In determining which data fields in the relational schemas of our supermarket management database should be indexed or hashed, it's essential to focus on fields that are frequently accessed or queried to enhance performance and operational efficiency. For instance, the primary keys such as *pk_product_id* in **t_products**, *pk_variant_id* in **t_product_variants**, and *pk_supplier_id* in **t_suppliers** should be indexed to facilitate rapid searches and efficient join operations across tables. Similarly, indexing foreign keys like *fk_product_id* in **t_product_variants** and *fk_supplier_id* in **t_supplier_contacts** will significantly improve the performance of queries that involve multiple tables. In the Product Sales System, fields such as *pk_transaction_id* in **t_transaction_records** and *fk_customer_id*, *fk_promotion_id* should also be indexed due to their frequent use in transactional queries and analytics. For the Employee Management System, *pk_employee_id* in **t_employees** must be indexed to ensure quick access to employee details. As for hashing, it is generally reserved for securing sensitive data that does not involve direct retrieval operations, such as passwords, which are currently not part of the presented schemas. If fields requiring enhanced security are introduced in future revisions, hashing should be considered for those fields. Additionally, if the system demands efficient text searches, such as searching products or companies by names like *product_name* or *company_name*, considering full-text indexing might prove beneficial to improve search capabilities and response times. Overall, the strategic application of indexing on these key fields will ensure that our database remains responsive and efficient, even under significant operational loads.

2.5 Simulated Data Generation

In the appendix of our report, we provide a comprehensive collection of realistic data that exemplifies the functioning of our supermarket management database system. This data is meticulously crafted to simulate the actual operations of a supermarket, reflecting the complexity and variability of real-world scenarios. The inclusion of such data serves multiple purposes: it allows stakeholders to visualize how the system manages and processes information, supports the validation and testing of the database functionalities, and aids in training users on the system by providing them with practical examples.

These realistic datasets include a variety of entries across different tables, such as product listings, supplier details, transaction records, and employee profiles. Each entry is designed to mirror typical supermarket data in terms of structure, volume, and interdependencies. This attention to detail ensures that the test scenarios are as close to actual supermarket operations as possible, which is crucial for accurately assessing the system's performance and reliability.

3. SQL Quires and Implementation

Our back-end implementation comprises five distinct parts, spanning from sections 1.1 to 5.12, which collectively include 38 SQL scripts. These scripts are categorized according to the subsystem they pertain to within our database architecture: Employee Management System (1.1-1.6), Product Sales System (2.1-2.7), Product Supply System (3.1-3.6), Product Organization System (4.1-4.7), and display functionalities for tables (5.1-5.12). Given the constraints on document length, we focus on explicating the core functionalities integral to our system operations.The full code has been attached to the appendix.

In the context of the Employee Management System, function 1.4 is crucial for handling employee dismissals. To maintain data integrity and continuity, dismissed employees are not removed from the system. Instead, their records are updated to reflect their status change, which prevents any loss of associated sales data. The SQL command for this operation is as follows:

```
UPDATE t_employees
SET
    job_position = 'dismission',
    salary = 0,
    work_schedules = 'No plan'
WHERE pk_employee_id = ?;
```

For the Product Sales System, functions 2.5, 2.6, and 2.7 manage the creation of transaction details. This process starts with the establishment of an empty transaction record, followed by the generation of transaction details based on purchased goods, and culminates with the aggregation of these details to compute the total transaction price.

In the Product Supply System, function 3.4 addresses the replenishment of stock. It involves updating the quantity of goods in the warehouse and subsequently documenting this change through a supply record.

For the Product Organization System, function 4.5 ensures that shelf quantities are adequately maintained. This is achieved by first decreasing the warehouse stock and then increasing the corresponding shelf quantity to rectify any deficiencies.

Lastly, function 5.1 serves a crucial role in displaying comprehensive information from the t_customer table, facilitating straightforward access to customer data.

These queries serve crucial functions in maintaining the operational efficiency of the supermarket, ranging from inventory management and sales tracking to employee management and supplier relations. They ensure that the database system supports real-time updates and data-driven decision-making, enhancing both customer satisfaction and business operations. In the forthcoming sections of our report, we will illustrate the tangible impacts that these SQL functions have on the user interface and overall user experience of our supermarket management database system. Given the constraints of document length and space, we will selectively present screenshots that exemplify the operation of some of the key SQL functions.

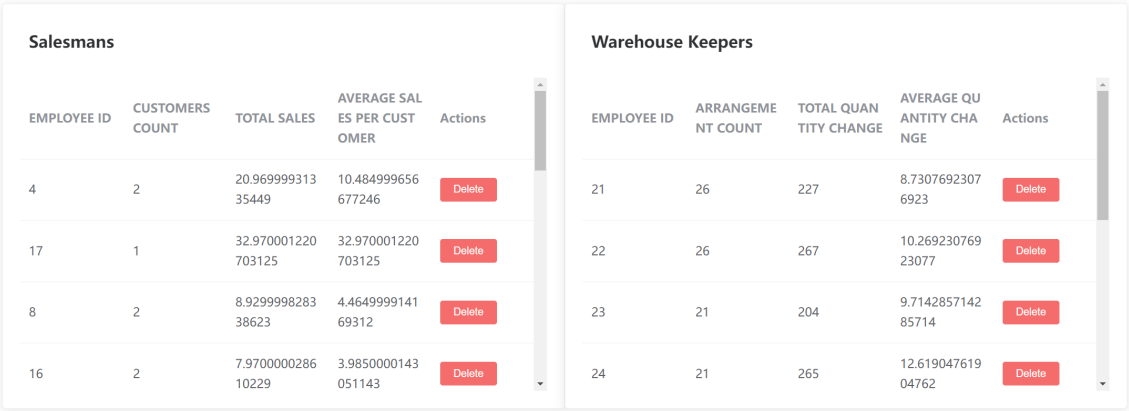

Figure 4: Screenshot of employee deletion

3.1 SQL Implementation of Practical Daily Operations and Activities

Fig.4 illustrates a specific functionality within the frontend interface of the supermarket management database system, focusing on the capability to delete employee records. This function is part of both the "Salesman" and "Warehouse Keepers" sections of the system, allowing for the management of employee data directly from the interface.

The screenshot (Fig.5) displays two sections of the frontend interface of the supermarket management database system, designed for adding new products and their variants into the database. This interface is crucial for maintaining an up-to-date inventory and allowing for the expansion of product offerings.

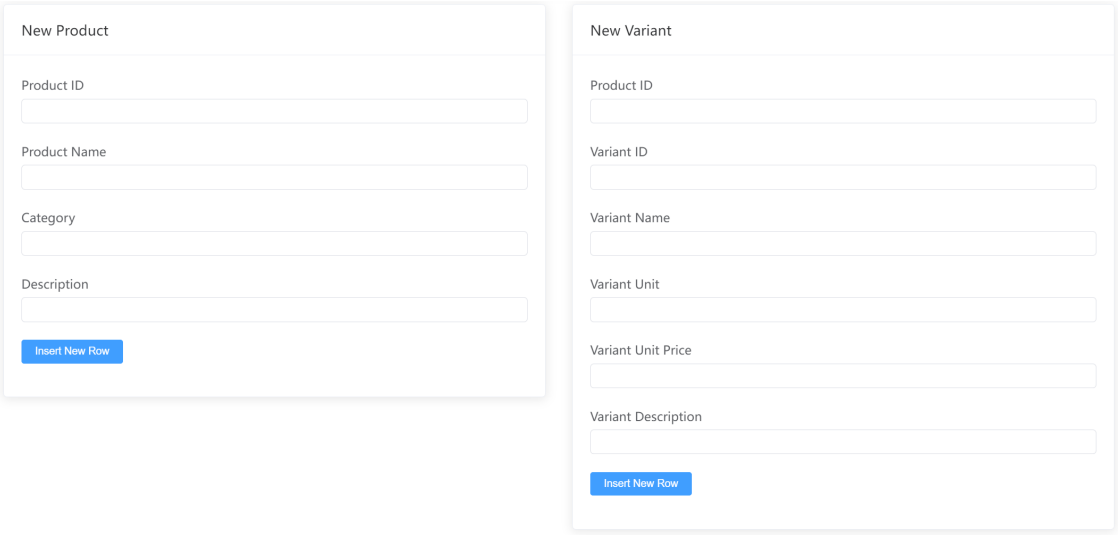

Figure 5: Screenshot of product and variant insertion

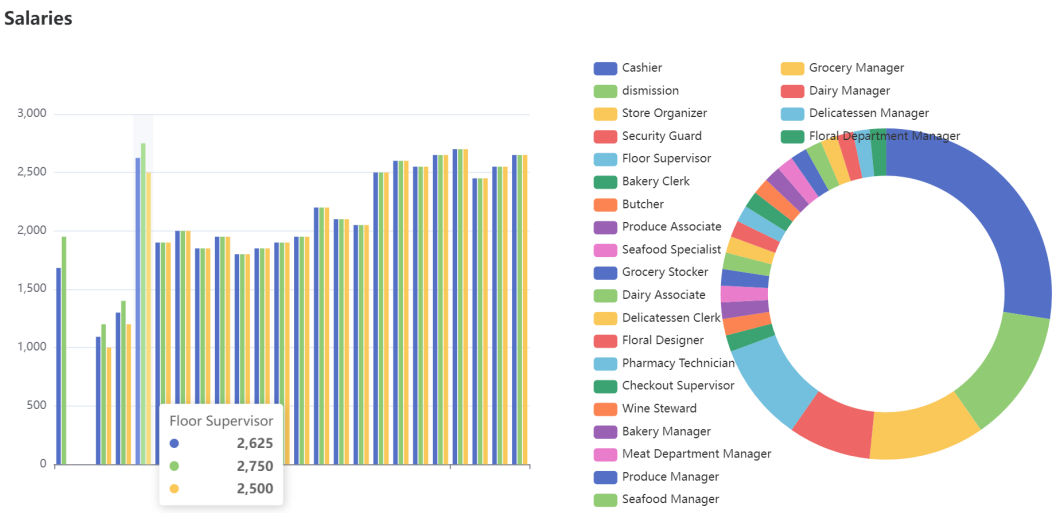

Figure 6: SQL queries implementation which are of an analytic or data mining nature

3.2 SQL Implementation of Analytic and Data Mining

In our supermarket management database system, we extend the utility of SQL to include sophisticated data analysis and data mining capabilities. These functionalities are critical for extracting valuable insights from the vast amounts of data accumulated through daily operations. As the Fig.6 shows. We provide salary comparisons for different employees and proportional distribution of different employees. More in this section. For example, the detailed code of SQL and how to connect to the MySQL server will be introduced in detail in the appendix.

4. Database Refinement via Large Language Models

We energized our database by applying modern Large Language Model to it.

Our advanced database system is equipped to process natural language queries by

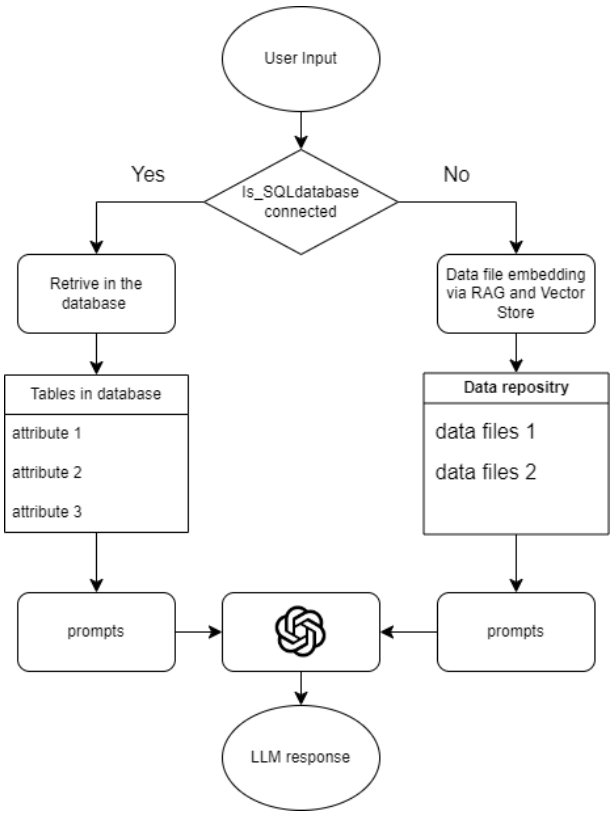

Figure 7: Flowchart of our advanced database system

first determining the data source—either an SQL database or text files. For data stored in SQL databases, the system directly retrieves the required information. Conversely, for text files, the system employs Retrieval Augmented Generation (RAG) and Vector Store technologies to facilitate data embedding. These processes generate precise prompts that are then interpreted by a sophisticated Language Model. This model is capable of providing real-time and accurate responses to user queries.

Fig.8 demonstrates the proficiency of our smart database in managing a diverse range of queries within the context of a supermarket management system. This includes identifying customers with the highest account balances, detailing product descriptions, enumerating product variants and their pricing, as well as generating comprehensive reports on weekly sales and revenue for the year 2023. This showcases the system's extensive capabilities in delivering efficient and reliable data handling and analytics.

5. Conclusion and Self-Evaluation

Our project, centered on developing a sophisticated supermarket management database system, has successfully culminated in a platform that integrates advanced SQL capabilities with cutting-edge technologies like Retrieval-Augmented Generation (RAG), Vector Store, and a Generative Pre-trained Transformer (GPT) for natural language processing. This integration has enabled the system to handle a wide range of functionalities crucial for effective supermarket management, including inventory tracking, sales analysis,

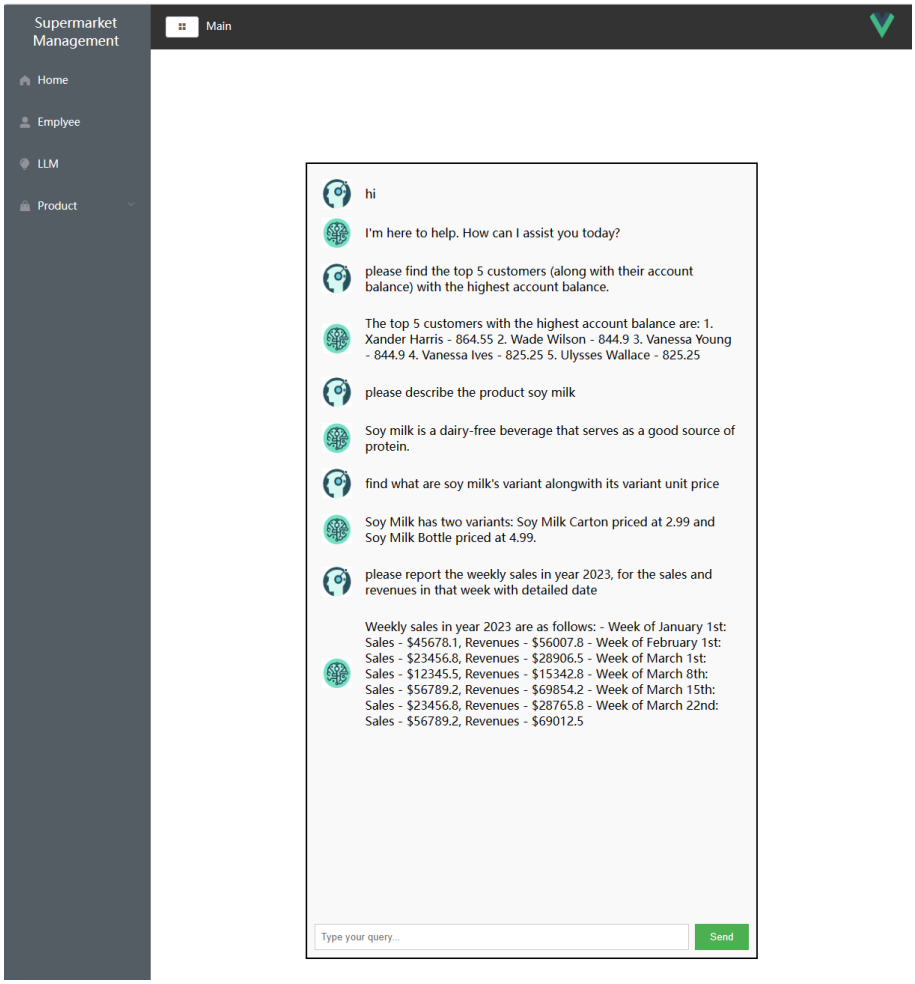

Figure 8: Demonstration example of the smart database

employee management, and customer relationship management.

The system's ability to process both structured SQL queries and unstructured natural language inputs significantly enhances its usability and accessibility, making it adaptable to various user competencies. Furthermore, the implementation of technologies like RAG and Vector Store ensures that the system remains robust and efficient, even when handling large datasets or complex query requirements. This is evident in the system's performance in generating detailed reports and insights, such as weekly sales trends, top customer identification, and comprehensive product information management.

6. Work Division of the Project

Task	Contributors
ER diagram and relation schema	Ziyu Xie, Yifei Dai
SQL and backend implementation	Tengfei Ma, Tieding Ma
Frontend implementation	You Lv
Crafting prompts using LLM	Kaiying Han
Report writing	Sixuan Mao

7. Appendices

```
// this is core backend file, the whole code is at https://github.com/Tengfei-Ma13206/CSC3170proj/tree/backend20240420
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(bodyParser.json());

// create connection
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '013206',
  database: 'comprehensive_supermarket',
  multipleStatements: true
});

// test the connection
connection.connect(err => {
    if (err) throw err;
    console.log('Connected to the database successfully!');
});

// 1.1
// query salesman info
app.get('/salesman', (req, res) => {
    console.log('Request received for /salary');
    const query = `
        SELECT
            e.pk_employee_id AS employee_id,
            COUNT(c.pk_customer_id) AS customers_count,
            SUM(td.discounted_total_price) AS total_sales,
            AVG(td.discounted_total_price) AS average_sales_per_customer
        FROM
            t_employees e
            JOIN t_transaction_records tr ON e.pk_employee_id = tr.fk_responsible_employee_id
            JOIN t_transaction_details td ON tr.pk_transaction_id = td.pk_transaction_id
            JOIN t_customers c ON tr.fk_customer_id = c.pk_customer_id
        GROUP BY
            e.pk_employee_id;
    `;
```

```
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 1.2
// query warehouse keeper info
    app.get('/warehouseKeeper', (req, res) => {
        const query = `
            SELECT
                e.pk_employee_id AS employee_id,
                COUNT(*) AS arrangement_count,
                SUM(ABS(pmr.warehouse_change_quantity)) AS total_quantity_change,
                AVG(ABS(pmr.warehouse_change_quantity)) AS average_quantity_change
            FROM
                t_employees e
                JOIN t_product_management_records pmr ON e.pk_employee_id = pmr.fk_responsible_employee_id
            GROUP BY
                e.pk_employee_id;
        `;
        connection.query(query, (err, results) => {
            if (err) {
                return res.status(500).send('Failed to execute query: ' + err.message);
            }
            res.json(results);
        });
    });
// 1.3
// query expense of employees
app.get('/salary', (req, res) => {
    console.log('Request received for /salary');
    const query = `
        SELECT
            e.job_position AS job_category,
            COUNT(*) AS employee_count,
            AVG(e.salary) AS average_salary,
            MAX(e.salary) AS highest_salary,
            MIN(e.salary) AS lowest_salary
        FROM
            t_employees e
        GROUP BY
            e.job_position;
```

```
88      ';
89      connection.query(query, (err, results) => {
90          if (err) {
91              return res.status(500).send('Failed to execute query: ' +
    err.message);
92          }
93          res.json(results);
94      });
95  });
96
97  // 1.4
98  // dismiss an employee, due to foreighn keys, we mark the employee as
    dismission instead of removing directly.
99  // we do not want the sales related to the dismissed employee to
    disappear.
100 app.post('/dismission', (req, res) => {
101     const { employee_id } = req.body;
102     const updateQuery = `
103         UPDATE t_employees
104         SET
105             job_position = 'dismission',
106             salary = 0,
107             work_schedules = 'No plan'
108         WHERE pk_employee_id = ?;
109     `;
110     connection.query(updateQuery, [employee_id], (err, result) => {
111         if (err) {
112             return res.status(500).send('Failed to update employee
    record: ' + err.message);
113         }
114         if (result.affectedRows === 0) {
115             return res.status(404).send({ message: 'No employee found
    with the given ID' });
116         }
117         res.send({ message: 'Employee dismission updated successfully'
    });
118     });
119 });
120
121 // 1.5
122 // add new employee
123 app.post('/addEmployee', (req, res) => {
124     console.log('Request received for /addEmployee');
125     const { pk_employee_id, job_position, salary, phone_number,
    work_schedules } = req.body;
126
127     const query = `
128         USE comprehensive_supermarket;
129         INSERT INTO t_employees (
```

```
            pk_employee_id,
            job_position,
            salary,
            phone_number,
            work_schedules
        )
        VALUES (?, ?, ?, ?, ?);
    `;

    connection.query(query, [pk_employee_id, job_position, salary, phone_number, work_schedules], (err, result) => {
      if (err) {
        return res.status(500).send('Failed to add new employee: ' + err.message);
      }
      res.send({ message: `New employee with ID ${pk_employee_id} added successfully` });
    });
  });

// 1.6
// check salary for employees
app.get('/personalSalary/:employeeId', (req, res) => {
    const employeeId = req.params.employeeId;

    const query = `
        SELECT
            e.pk_employee_id AS employee_id,
            e.salary,
            e.work_schedules AS work_schedule
        FROM
            t_employees e
        WHERE
            e.pk_employee_id = ?;
    `;

    connection.query(query, [employeeId], (err, results) => {
        if (err) {
            return res.status(500).send('Failed to retrieve salary information: ' + err.message);
        }
        if (results.length > 0) {
            res.json(results[0]);
        } else {
            res.status(404).send('Employee not found');
        }
    });
});

```

```
175 // 2.1
176 // check category of products
177 app.get('/categorySales', (req, res) => {
178     const query = `
179         SELECT
180             p.category,
181             SUM(td.purchasing_quantity) AS total_quantity,
182             SUM(td.purchasing_quantity * td.discounted_total_price) AS
    total_sales,
183             SUM(td.purchasing_quantity * (td.discounted_total_price - sr
    .total_price / sr.supply_quantity)) AS total_profit,
184             AVG(td.discounted_total_price / td.purchasing_quantity) AS
    average_selling_price
185         FROM
186             t_products p
187             JOIN t_product_variants pv ON p.pk_product_id = pv.
    pk_product_id
188             JOIN t_transaction_details td ON pv.pk_product_id = td.
    fk_product_id AND pv.pk_variant_id = td.fk_variant_id
189             JOIN t_supply_records sr ON pv.pk_product_id = sr.
    fk_supply_product_id AND pv.pk_variant_id = sr.fk_supply_variant_id
190         GROUP BY
191             p.category;
192     `;
193     connection.query(query, (err, results) => {
194         if (err) {
195             return res.status(500).send('Failed to execute query: ' +
    err.message);
196         }
197         res.json(results);
198     });
199 });
200
201 // 2.2
202 // create an account for customer. Every customer who buy something here
     must have an account first.
203 app.post('/createCustomer',(req,res) => {
204     const { pk_customer_id, customer_name, phone_number, account_balance
    } = req.body;
205     const query = `
206         INSERT INTO t_customers(
207             pk_customer_id,
208             customer_name,
209             phone_number,
210             account_balance
211         )
212         VALUES(?, ?, ?, ?);
213     `;
214     connection.query(query, [pk_customer_id, customer_name, phone_number
```

```
    , account_balance], (err, result) => {
215         if (err) {
216           // fail
217           return res.status(500).send('Failed to add new customer: ' +
    err.message);
218         }
219         // success
220         res.send({ message: `New customer with ID ${pk_customer_id}
    added successfully` });
221       });
222 });
223
224 // 2.3
225 app.post('/createPromotion',(req,res) => {
226     const { pk_promotion_id, threshold_amount, discount_amount,
    start_date, end_date } = req.body;
227     const query = `
228         INSERT INTO t_reduction_promotions(
229             pk_promotion_id,
230             threshold_amount,
231             discount_amount,
232             start_date,
233             end_date
234         )
235         VALUES (?, ?, ?, ?, ?);
236     `
237     connection.query(query,[pk_promotion_id, threshold_amount,
    discount_amount, start_date, end_date],(err, result) => {
238         if (err) {
239             return res.status(500).send('Failed to add new promotion: '
    + err.message);
240         }
241         res.send({ message: `New promotion with ID ${pk_promotion_id}
    added successfully` });
242     });
243 });
244
245 // 2.4
246 app.post('/addVariantDiscount', (req, res) => {
247     const { variant_discount_id, discount_product_id,
    discount_variant_id, variant_discount_rate, start_date, end_date } =
    req.body;
248
249     const sqlQuery = `
250         INSERT INTO t_variant_discounts(
251             pk_variant_discount_id,
252             fk_discount_product_id,
253             fk_discount_variant_id,
254             variant_discount_rate,
```

```
            start_date,
            end_date
        )
        VALUES (?, ?, ?, ?, ?, ?);
    `;

    // Execute the query
    connection.query(sqlQuery, [variant_discount_id, discount_product_id, discount_variant_id, variant_discount_rate, start_date, end_date], (err, results) => {
        if (err) {
            return res.status(500).send({ error: 'Failed to add variant discount: ' + err.message });
        }
        res.send({ message: 'Variant discount added successfully' });
    });
});

// 2.5
app.post('/createTransRecord',(req,res) => {
    const { pk_transaction_id, fk_responsible_employee_id, fk_customer_id, transaction_way, fk_promotion_id } = req.body;
    const query = `
        INSERT INTO t_transaction_records(
            pk_transaction_id,
            fk_responsible_employee_id,
            fk_customer_id,
            transaction_date,
            transaction_time,
            transaction_way,
            fk_promotion_id,
            initial_amount,
            discounted_amount
        )
        VALUES
        (
            ?, ?, ?,
            DATE(NOW()),
            TIME(NOW()),
            ?, ?, 0, 0
        );
    `;
    connection.query(query,[pk_transaction_id, fk_responsible_employee_id, fk_customer_id, transaction_way, fk_promotion_id],(err, result) => {
        if (err) {
            return res.status(500).send('Failed to add new Transaction Record: ' + err.message);
        }
```

```
297         res.send({ message: `New Transaction Record with ID ${
    pk_transaction_id} added successfully` });
298     });
299 });
300
301 // 2.6
302 app.post('/addTransDetails', (req, res) => {
303     const { pk_transaction_id, pk_transaction_detail_id, fk_product_id,
    fk_variant_id, purchasing_quantity } = req.body;
304
305
306     connection.beginTransaction(err => {
307         if (err) {
308             return res.status(500).send('Transaction failed to start: '
    + err.message);
309         }
310
311         connection.query(`
312             SELECT start_date, end_date, variant_discount_rate,
    shelf_quantity, variant_unit_price
313             FROM t_variant_discounts
314             JOIN t_product_variants ON (t_variant_discounts.
    fk_discount_product_id = t_product_variants.pk_product_id AND
    t_variant_discounts.fk_discount_variant_id = t_product_variants.
    pk_variant_id)
315             WHERE fk_discount_product_id = ? AND fk_discount_variant_id
    = ?;`,
316             [fk_product_id, fk_variant_id],
317             (err, results) =>
318             {
319                 if (err || results === 0)
320                 {
321                     connection.rollback
322                     (() =>
323                         {
324                             return res.status(500).send('Error fetching
    discounts or products: ' + (err ? err.message : 'No data found.'));
325                         }
326                     );
327                 }
328
329                 const { start_date, end_date, variant_discount_rate,
    shelf_quantity, variant_unit_price } = results[0];
330                 const current_date = new Date();
331
332                 if (shelf_quantity < purchasing_quantity)
333                 {
334                     connection.rollback
335                     (
```

```
                        () =>
                        {
                            return res.status(400).send('Insufficient
stock.');
                        }
                    )
                }
                else
                {
                    const isDiscountPeriod = current_date >= start_date
&& current_date <= end_date;
                    const discounted_total_price = purchasing_quantity *
 variant_unit_price * (isDiscountPeriod ? (1 - variant_discount_rate)
 : 1);

                    connection.query
                    (`
                        INSERT INTO t_transaction_details
                        (
                            pk_transaction_id,
                            pk_transaction_detail_id,
                            fk_product_id,
                            fk_variant_id,
                            purchasing_quantity,
                            discounted_total_price
                        )
                        VALUES (?,?,?,?,?,?);
                        UPDATE t_product_variants
                        SET shelf_quantity = shelf_quantity - ?
                        WHERE pk_product_id = ? AND pk_variant_id = ?;
                    `,
                    [
                        pk_transaction_id, pk_transaction_detail_id,
                        fk_product_id, fk_variant_id,
purchasing_quantity,
                        discounted_total_price,
                        purchasing_quantity, fk_product_id,
fk_variant_id
                    ],
                    (err, results) =>
                    {
                        if (err)
                        {
                            connection.rollback
                            (() =>
                            {
                                return res.status(500).send('Failed to
insert transaction details: ' + err.message);
                            });
```

```
                        }
                        else
                        {
                            connection.commit
                            (err =>
                            {
                                if (err)
                                {
                                    connection.rollback
                                    (() =>
                                    {
                                        return res.status(500).send('Transaction commit failed: ' + err.message);
                                    });
                                }
                                res.send({ message: 'Transaction details added successfully' });
                            });
                        }
                    }
                    )
                }
            });
    });
});


// 2.7
app.patch('/updateTransRecords', (req, res) => {
    const { pk_transaction_id, fk_promotion_id } = req.body;

    connection.beginTransaction(err => {
        if (err) {
            return res.status(500).send('Transaction failed to start: ' + err.message);
        }

        // calculate total price
        connection.query(`
            SELECT SUM(discounted_total_price) AS total_price
            FROM t_transaction_details
            WHERE pk_transaction_id = ?`,
            [pk_transaction_id],
            (err, results) => {
                if (err) {
                    connection.rollback(() => {
                        return res.status(500).send('Error fetching transaction details: ' + err.message);
                    });
```

```
423                 }
424
425                 let sum_price = results[0].total_price;
426
427                 // look up promotion info
428                 connection.query(`
429                     SELECT start_date, end_date, threshold_amount,
    discount_amount
430                     FROM t_reduction_promotions
431                     WHERE pk_promotion_id = ?`,
432                     [fk_promotion_id],
433                     (err, promoResults) => {
434                         if (err) {
435                             connection.rollback(() => {
436                                 return res.status(500).send('Error
    fetching promotion details: ' + err.message);
437                             });
438                         }
439
440                         if (promoResults.length === 0) {
441                             connection.rollback(() => {
442                                 return res.status(404).send('Promotion
    not found.');
443                             });
444                         }
445
446                         const { start_date, end_date, threshold_amount,
    discount_amount } = promoResults[0];
447                         const currentDate = new Date();
448                         let discounted_amount = sum_price;
449
450                         // check conditions of promotion
451                         if (currentDate >= start_date && currentDate <=
    end_date && sum_price >= threshold_amount) {
452                             discounted_amount = sum_price -
    discount_amount;
453                         }
454
455                         // update the transaction record
456                         connection.query(`
457                             UPDATE t_transaction_records
458                             SET initial_amount = ?, discounted_amount =
    ?
459                             WHERE pk_transaction_id = ?`,
460                             [sum_price, discounted_amount,
    pk_transaction_id],
461                             (err, updateResults) => {
462                                 if (err) {
463                                     connection.rollback(() => {
```

```
                                    return res.status(500).send('Failed to update transaction records: ' + err.message);
                                });
                            }
                            connection.commit(err => {
                                if (err) {
                                    connection.rollback(() => {
                                        return res.status(500).send('Transaction commit failed: ' + err.message);
                                    });
                                }
                                res.send({ message: 'Transaction records finalized successfully' });
                            });
                        }
                    );
                }
            );
        }
    );
});

// 3.1
app.get('/productSupply', (req, res) => {
    const sql = `
        SELECT
            p.pk_product_id AS product_id,
            p.product_name,
            s.pk_supplier_id AS supplier_id,
            sr.supply_quantity,
            sr.total_price,
            sr.total_price / sr.supply_quantity AS unit_price
        FROM
            t_products p
            JOIN t_supply_records sr ON p.pk_product_id = sr.fk_supply_product_id
            JOIN t_suppliers s ON sr.fk_supplier_id = s.pk_supplier_id
        WHERE
            p.pk_product_id IN (
                SELECT fk_supply_product_id
                FROM t_supply_records
                GROUP BY fk_supply_product_id
                HAVING COUNT(DISTINCT fk_supplier_id) >= 2
            )
        ORDER BY
            p.pk_product_id,
            s.pk_supplier_id;
    `;
```

```
509
510     connection.query(sql, (err, results) => {
511         if (err) {
512             return res.status(500).send({ error: 'Failed to fetch
    products: ' + err.message });
513         }
514         res.send(results);
515     });
516 });
517
518 // 3.2
519 app.get('/supplier', (req, res) => {
520     const sqlQuery = `
521         SELECT
522             s.pk_supplier_id AS supplier_id,
523             s.company_name,
524             COUNT(DISTINCT sr.fk_supply_variant_id) AS variant_count,
525             SUM(sr.supply_quantity) AS total_supply_quantity,
526             AVG(sr.supply_quantity) AS avg_supply_quantity,
527             AVG(sr.total_price / sr.supply_quantity) AS avg_unit_price
528         FROM
529             t_suppliers s
530             JOIN t_supply_records sr ON s.pk_supplier_id = sr.
    fk_supplier_id
531         GROUP BY
532             s.pk_supplier_id,
533             s.company_name;
534     `;
535
536     connection.query(sqlQuery, (err, results) => {
537         if (err) {
538             return res.status(500).send({ error: 'Failed to fetch
    supplier statistics: ' + err.message });
539         }
540         res.send(results);
541     });
542 });
543
544 // 3.3
545 app.get('/warehouse', (req, res) => {
546     const sqlQuery = `
547         SELECT
548             p.pk_product_id AS product_id,
549             AVG(pv.warehouse_quantity) OVER (PARTITION BY p.category) AS
    avg_warehouse_quantity,
550             pv.warehouse_quantity AS current_warehouse_quantity,
551             CASE
552                 WHEN pv.warehouse_quantity < AVG(pv.warehouse_quantity)
    OVER (PARTITION BY p.category) THEN 'Yes'
```

```
                ELSE 'No'
            END AS need_to_update,
            s.pk_supplier_id AS supplier_id,
            s.company_name,
            sc.phone_number AS supplier_phone_number
        FROM
            t_products p
            JOIN t_product_variants pv ON p.pk_product_id = pv.
    pk_product_id
            JOIN t_supply_records sr ON pv.pk_variant_id = sr.
    fk_supply_variant_id
            JOIN t_suppliers s ON sr.fk_supplier_id = s.pk_supplier_id
            JOIN t_supplier_contacts sc ON s.pk_supplier_id = sc.
    fk_supplier_id
        ORDER BY
            CASE
                WHEN pv.warehouse_quantity < AVG(pv.warehouse_quantity)
    OVER (PARTITION BY p.category) THEN 0
                ELSE 1
            END,
            p.pk_product_id;
    `;

    connection.query(sqlQuery, (err, results) => {
        if (err) {
            return res.status(500).send({ error: 'Failed to fetch
    product warehouse statistics: ' + err.message });
        }
        res.send(results);
    });
});

// 3.4
app.post('/updateWarehouse', (req, res) => {
    const {
        product_id,
        variant_id,
        quantity,
        supplier_id,
        purchase_price,
        supply_record_id,
        supply_date,
        pay_term
    } = req.body;

    connection.beginTransaction(err => {
        if (err) {
            return res.status(500).send('Transaction failed to start: '
    + err.message);
```

```
        }

        connection.query(`
            UPDATE t_product_variants
            SET warehouse_quantity = warehouse_quantity + ?
            WHERE pk_product_id = ? AND pk_variant_id = ?`,
            [quantity, product_id, variant_id],
            (err, result) => {
                if (err) {
                    connection.rollback(() => {
                        return res.status(500).send('Failed to update warehouse quantity: ' + err.message);
                    });
                }

                connection.query(`
                    INSERT INTO t_supply_records (pk_supply_record_id, fk_supply_product_id, fk_supply_variant_id, fk_supplier_id, supply_date, supply_quantity, total_price, pay_term)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?)`,
                    [supply_record_id, product_id, variant_id, supplier_id, supply_date, quantity, quantity * purchase_price, pay_term],
                    (err, result) => {
                        if (err) {
                            connection.rollback(() => {
                                return res.status(500).send('Failed to insert supply record: ' + err.message);
                            });
                        }
                        connection.commit(err => {
                            if (err) {
                                connection.rollback(() => {
                                    return res.status(500).send('Transaction commit failed: ' + err.message);
                                });
                            }
                            res.send({ message: 'Supply updated successfully' });
                        });
                    }
                );
            }
        );
    });
});

// 3.5
app.post('/createSupplier', (req, res) => {
```

```
637     const { supplier_id, company_name, company_address, company_website
    } = req.body;
638 
639     const sqlQuery = `
640         INSERT INTO t_suppliers (pk_supplier_id, company_name,
    company_address, company_website)
641         VALUES (?, ?, ?, ?);
642     `;
643 
644     connection.query(sqlQuery, [supplier_id, company_name,
    company_address, company_website], (err, results) => {
645         if (err) {
646             return res.status(500).send({ error: 'Failed to create
    supplier: ' + err.message });
647         }
648         res.send({ message: 'Supplier created successfully', supplierId:
     supplier_id });
649     });
650 });
651 
652 // 3.6
653 app.post('/addSupplierContacts', (req, res) => {
654     const { contact_id, supplier_id, contact_name, contact_title,
    phone_number, email } = req.body;
655 
656     const sqlQuery = `
657         INSERT INTO t_supplier_contacts (pk_contact_id, fk_supplier_id,
    contact_name, contact_title, phone_number, email)
658         VALUES (?, ?, ?, ?, ?, ?);
659     `;
660 
661     connection.query(sqlQuery, [contact_id, supplier_id, contact_name,
    contact_title, phone_number, email], (err, results) => {
662         if (err) {
663             return res.status(500).send({ error: 'Failed to add supplier
     contact: ' + err.message });
664         }
665         res.send({ message: 'Supplier contact added successfully',
    contactId: contact_id });
666     });
667 });
668 // 4.1
669 app.get('/allProductInfo', (req, res) => {
670     const sqlQuery = `
671         SELECT
672             p.product_name AS product_name,
673             p.category AS category,
674             pv.pk_variant_id AS variant_id,
675             pv.variant_unit AS variant_unit,
```

```
            pv.variant_unit_price AS unit_price
        FROM
            t_products p
        JOIN
            t_product_variants pv ON p.pk_product_id = pv.pk_product_id;
    `;

    connection.query(sqlQuery, (err, results) => {
        if (err) {
            return res.status(500).send({ error: 'Failed to fetch product information: ' + err.message });
        }
        res.send(results);
    });
});

// 4.2
app.get('/checkReplenishment', (req, res) => {
    const sqlQuery = `
        SELECT
            p.product_name,
            mag_product.pk_variant_id,
            p.category,
            mag_product.shelf_quantity AS shelf_quantity,
            AVG(mag_product.shelf_quantity) OVER (PARTITION BY p.category) AS avg_shelf_quantity,
            mag_product.warehouse_quantity AS warehouse_quantity,
            CASE
                WHEN mag_product.shelf_quantity < AVG(mag_product.shelf_quantity) OVER (PARTITION BY p.category) THEN 'yes'
                ELSE 'no'
            END AS whether_to_update_shelf_quantity
        FROM
            t_products p
        JOIN
            (SELECT pk_product_id, pk_variant_id, shelf_quantity, warehouse_quantity FROM t_product_variants) mag_product ON p.pk_product_id = mag_product.pk_product_id
        ORDER BY
            CASE WHEN whether_to_update_shelf_quantity = 'yes' THEN 0 ELSE 1 END,
            p.category,
            p.product_name;
    `;

    connection.query(sqlQuery, (err, results) => {
        if (err) {
            console.error('Failed to fetch product inventory details: ' + err.message);
```

```
718             return res.status(500).send({ error: 'Database query failed'
    });
719         }
720         res.send(results);
721     });
722 });
723
724 // 4.3
725 app.post('/createProduct', (req, res) => {
726     const { product_id, product_name, category, description } = req.body
    ;
727
728     const sqlQuery = `
729         INSERT INTO t_products (pk_product_id, product_name, category,
    description)
730         VALUES (?, ?, ?, ?);
731     `;
732
733     connection.query(sqlQuery, [product_id, product_name, category,
    description], (err, results) => {
734         if (err) {
735             return res.status(500).send({ error: 'Failed to create
    product: ' + err.message });
736         }
737         res.send({ message: 'Product created successfully', productId:
    product_id });
738     });
739 });
740
741 // 4.4
742 app.post('/createVariant', (req, res) => {
743     const { product_id, variant_id, variant_name, variant_unit,
    variant_unit_price, variant_description } = req.body;
744
745     const sqlQuery = `
746         INSERT INTO t_product_variants (pk_product_id, pk_variant_id,
    variant_name, variant_unit, variant_unit_price, variant_description,
    warehouse_quantity, shelf_quantity)
747         VALUES (?, ?, ?, ?, ?, ?, 0, 0);
748     `;
749     connection.query(sqlQuery, [product_id, variant_id, variant_name,
    variant_unit, variant_unit_price, variant_description ], (err,
    results) => {
750         if (err) {
751             return res.status(500).send({ error: 'Failed to create
    product variant: ' + err.message });
752         }
753         res.send({ message: 'Product variant created successfully',
    variantId: variant_id });
```

```
    });
});
// 4.5
app.post('/updateShelf', (req, res) => {
    const { pk_management_record_id, responsible_employee_id, product_id, variant_id, warehouse_change_quantity, shelf_change_quantity } = req.body;

    // start transaction
    connection.beginTransaction(err => {
        if (err) {
            return res.status(500).send('Transaction failed to start: ' + err.message);
        }

        // insert  one piece of product management record
        const insertRecordSql = `
            INSERT INTO t_product_management_records
            (pk_management_record_id, management_date, management_time, fk_responsible_employee_id, fk_product_id, fk_variant_id, warehouse_change_quantity, shelf_change_quantity)
            VALUES
            (?, CURDATE(), CURTIME(), ?, ?, ?, ?, ?);
        `;
        connection.query(insertRecordSql, [pk_management_record_id, responsible_employee_id, product_id, variant_id, warehouse_change_quantity, shelf_change_quantity], (err, results) => {
            if (err) {
                connection.rollback(() => {
                    return res.status(500).send('Failed to insert product management record: ' + err.message);
                });
            }

            // update warehouse quantity
            const updateWarehouseSql = `
                UPDATE t_product_variants
                SET warehouse_quantity = warehouse_quantity + ?
                WHERE pk_product_id = ? AND pk_variant_id = ?;
            `;
            connection.query(updateWarehouseSql, [warehouse_change_quantity, product_id, variant_id], (err, results) => {
                if (err) {
                    connection.rollback(() => {
                        return res.status(500).send('Failed to update warehouse quantity: ' + err.message);
                    });
```

```
                }

                // update shelf_quantity
                const updateShelfSql = `
                    UPDATE t_product_variants
                    SET shelf_quantity = shelf_quantity + ?
                    WHERE pk_product_id = ? AND pk_variant_id = ?;
                `;
                connection.query(updateShelfSql, [shelf_change_quantity, product_id, variant_id], (err, results) => {
                    if (err) {
                        connection.rollback(() => {
                            return res.status(500).send('Failed to update shelf quantity: ' + err.message);
                        });
                    }

                    // if every on the right track, commit
                    connection.commit(err => {
                        if (err) {
                            connection.rollback(() => {
                                return res.status(500).send('Transaction commit failed: ' + err.message);
                            });
                        }
                        res.send({ message: 'Shelf and warehouse quantities updated successfully' });
                    });
                });
            });
        });
    });
});

// 4.6
app.post('/rmProduct', (req, res) => {
    const { product_id } = req.body;

    connection.beginTransaction(err => {
        if (err) {
            return res.status(500).send('Transaction failed to start: ' + err.message);
        }

        connection.query(`
            UPDATE t_products
            SET description = 'sold out'
            WHERE pk_product_id = ?;
        `, [product_id], (err, result) => {
```

```
            if (err) {
                connection.rollback(() => {
                    return res.status(500).send('Failed to update product description: ' + err.message);
                });
            }

            connection.query(`
                UPDATE t_product_variants
                SET warehouse_quantity = 0,
                    shelf_quantity = 0,
                    variant_description = "soldout"
                WHERE pk_product_id = ?;
            `, [product_id], (err, result) => {
                if (err) {
                    connection.rollback(() => {
                        return res.status(500).send('Failed to update product variants quantities: ' + err.message);
                    });
                }

                connection.commit(err => {
                    if (err) {
                        connection.rollback(() => {
                            return res.status(500).send('Transaction commit failed: ' + err.message);
                        });
                    }
                    res.send({ message: 'Product marked as sold out and quantities set to zero successfully' });
                });
            });
        });
    });
});

// 4.7
app.post('/rmVariant', (req, res) => {
    const { product_id, variant_id } = req.body;

    // Begin a transaction
    connection.beginTransaction(err => {
        if (err) {
            return res.status(500).send('Transaction failed to start: ' + err.message);
        }

        const updateQuery = `
            UPDATE t_product_variants
```

```
            SET warehouse_quantity = 0,
                shelf_quantity = 0,
                variant_description = 'soldout'
            WHERE pk_product_id = ? AND pk_variant_id = ?;
        `;

        connection.query(updateQuery, [product_id, variant_id], (err, result) => {
            if (err) {
                connection.rollback(() => {
                    return res.status(500).send('Failed to update variant status: ' + err.message);
                });
            }

            // Commit the transaction if all is well
            connection.commit(err => {
                if (err) {
                    connection.rollback(() => {
                        return res.status(500).send('Transaction commit failed: ' + err.message);
                    });
                }
                res.send({ message: 'Variant status updated to soldout successfully' });
            });
        });
    });
});



// show tables
// 5.1
app.get('/showCustomers', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_customers;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.2
app.get('/showEmployees', (req, res) => {
```

```
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_employees;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' +
    err.message);
        }
        res.json(results);
    });
});
// 5.3
app.get('/showProductManagement', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_product_management_records
    ;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' +
    err.message);
        }
        res.json(results);
    });
});
// 5.4
app.get('/showProductVariants', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_product_variants;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' +
    err.message);
        }
        res.json(results);
    });
});
// 5.5
app.get('/showProducts', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_products;
    `;
    connection.query(query, (err, results) => {
        if (err) {
```

```
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.6
app.get('/showReductionPromotions', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_reduction_promotions;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.7
app.get('/showSupplierContacts', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_supplier_contacts;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.8
app.get('/showSuppliers', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_suppliers;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.9
```

```
app.get('/showSupplyRecords', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_supply_records;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.10
app.get('/showTransDetails', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_transaction_details;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.11
app.get('/showTransRecords', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_transaction_records;
    `;
    connection.query(query, (err, results) => {
        if (err) {
            return res.status(500).send('Failed to execute query: ' + err.message);
        }
        res.json(results);
    });
});
// 5.12
app.get('/showVariantDiscounts', (req, res) => {
    console.log('Request received');
    const query = `
    SELECT * FROM comprehensive_supermarket.t_variant_discounts;
    `;
    connection.query(query, (err, results) => {
        if (err) {
```

```
            return res.status(500).send('Failed to execute query: ' +
    err.message);
        }
        res.json(results);
    });
});


// end


const port = 3000;
app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Listing 1: Server configuration and API implementation

```
-- this is core backend file, the whole code is at https://github.com/
    Tengfei-Ma13206/CSC3170proj/tree/main/sql_
-- for first system
-- 1
SELECT
    e.pk_employee_id AS employee_id,
    COUNT(c.pk_customer_id) AS customers_count,
    SUM(td.discounted_total_price) AS total_sales,
    AVG(td.discounted_total_price) AS average_sales_per_customer
FROM
    t_employees e
    JOIN t_transaction_records tr ON e.pk_employee_id = tr.
    fk_responsible_employee_id
    JOIN t_transaction_details td ON tr.pk_transaction_id = td.
    pk_transaction_id
    JOIN t_customers c ON tr.fk_customer_id = c.pk_customer_id
GROUP BY
    e.pk_employee_id;

-- 2
SELECT
    e.pk_employee_id AS employee_id,
    COUNT(*) AS arrangement_count,
    SUM(ABS(pmr.warehouse_change_quantity)) AS total_quantity_change,
    AVG(ABS(pmr.warehouse_change_quantity)) AS average_quantity_change
FROM
    t_employees e
    JOIN t_product_management_records pmr ON e.pk_employee_id = pmr.
    fk_responsible_employee_id
GROUP BY
    e.pk_employee_id;

```

```sql
-- 3
SELECT
    e.job_position AS job_category,
    COUNT(*) AS employee_count,
    AVG(e.salary) AS average_salary,
    MAX(e.salary) AS highest_salary,
    MIN(e.salary) AS lowest_salary
FROM
    t_employees e
GROUP BY
    e.job_position;

-- 4
UPDATE t_employees
SET
    job_position = 'dismissal',
    salary = 0,
    work_schedules = "No plan"
WHERE pk_employee_id = 10;

-- 5
INSERT INTO t_employees (
    pk_employee_id,
    job_position,
    salary,
    phone_number,
    work_schedule
)
VALUES (
    10,
    'Sales Representative',
    5000.00,
    '123-456-7890',
    'Full-time'
);

-- 6
SELECT
    e.pk_employee_id AS employee_id,
    e.salary,
    e.work_schedules AS work_schedule
FROM
    t_employees e
WHERE
    e.pk_employee_id = 1;

-- for second system
SELECT
    p.category,
```

```
78      SUM(td.purchasing_quantity) AS total_quantity,
79      SUM(td.purchasing_quantity * td.discounted_total_price) AS
    total_sales,
80      SUM(td.purchasing_quantity * (td.discounted_total_price - sr.
    total_price / sr.supply_quantity)) AS total_profit,
81      AVG(td.discounted_total_price / td.purchasing_quantity) AS
    average_selling_price
82  FROM
83      t_products p
84      JOIN t_product_variants pv ON p.pk_product_id = pv.pk_product_id
85      JOIN t_transaction_details td ON pv.pk_product_id = td.fk_product_id
        and pv.pk_variant_id = td.fk_variant_id
86      JOIN t_supply_records sr ON pv.pk_product_id = sr.
    fk_supply_product_id and pv.pk_variant_id = sr.fk_supply_variant_id
87  GROUP BY
88      p.category;
89
90  -- for third system
91  -- 1
92  SELECT
93      p.pk_product_id AS product_id,
94      p.product_name,
95      s.pk_supplier_id AS supplier_id,
96      sr.supply_quantity,
97      sr.total_price,
98      sr.total_price / sr.supply_quantity AS unit_price
99  FROM
100     t_products p
101     JOIN t_supply_records sr ON p.pk_product_id = sr.
    fk_supply_product_id
102     JOIN t_suppliers s ON sr.fk_supplier_id = s.pk_supplier_id
103 WHERE
104     p.pk_product_id IN (
105         SELECT fk_supply_product_id
106         FROM t_supply_records
107         GROUP BY fk_supply_product_id
108         HAVING COUNT(DISTINCT fk_supplier_id) >= 2
109     )
110 ORDER BY
111     p.pk_product_id,
112     s.pk_supplier_id;
113
114 -- 2
115 SELECT
116     s.pk_supplier_id AS supplier_id,
117     s.company_name,
118     COUNT(DISTINCT sr.fk_supply_variant_id) AS variant_count,
119     SUM(sr.supply_quantity) AS total_supply_quantity,
120     AVG(sr.supply_quantity) AS avg_supply_quantity,
```

```
121     AVG(sr.total_price / sr.supply_quantity) AS avg_unit_price
122 FROM
123     t_suppliers s
124     JOIN t_supply_records sr ON s.pk_supplier_id = sr.fk_supplier_id
125 GROUP BY
126     s.pk_supplier_id,
127     s.company_name;
128
129 -- 3
130 SELECT
131     p.pk_product_id AS product_id,
132     AVG(pv.warehouse_quantity) OVER (PARTITION BY p.category) AS
        avg_warehouse_quantity,
133     pv.warehouse_quantity AS current_warehouse_quantity,
134     CASE
135         WHEN pv.warehouse_quantity < AVG(pv.warehouse_quantity) OVER (
        PARTITION BY p.category) THEN 'Yes'
136         ELSE 'No'
137     END AS need_to_update,
138     s.pk_supplier_id AS supplier_id,
139     s.company_name,
140     sc.phone_number AS supplier_phone_number
141 FROM
142     t_products p
143     JOIN t_product_variants pv ON p.pk_product_id = pv.pk_product_id
144     JOIN t_supply_records sr ON pv.pk_variant_id = sr.
        fk_supply_variant_id
145     JOIN t_suppliers s ON sr.fk_supplier_id = s.pk_supplier_id
146     JOIN t_supplier_contacts sc ON s.pk_supplier_id = sc.fk_supplier_id
147 ORDER BY
148     CASE
149         WHEN pv.warehouse_quantity < AVG(pv.warehouse_quantity) OVER (
        PARTITION BY p.category) THEN 0
150         ELSE 1
151     END,
152     p.pk_product_id;
153
154 -- 4
155 -- renew inventory
156 UPDATE t_product_variants
157 SET warehouse_quantity = warehouse_quantity + ?quantity
158 WHERE pk_product_id = ?product_id AND pk_variant_id = ?variant_id;
159
160 -- insert new record
161 INSERT INTO t_supply_records (pk_supply_record_id, fk_product_id,
        fk_variant_id, fk_supplier_id, supply_date, supply_quantity,
        purchase_price, total_price, pay_term)
162 VALUES (
163     ?supply_record_id,
```

```
    ?product_id,
    ?variant_id,
    ?supplier_id,
    ?supply_date,
    ?quantity,
    ?purchase_price,
    ?quantity * ?purchase_price,
    ?pay_term
);

-- 5
-- insert the new suppliers
INSERT INTO t_suppliers (pk_supplier_id, company_name, company_address,
    company_website)
VALUES (
    '111',
    'Sunrise Electronics',
    '123 Main Street, Anytown, USA',
    'www.sunriseelectronics.com'
);

-- insert the new supplier's contact
INSERT INTO t_supplier_contacts (pk_contact_id, fk_supplier_id,
    contact_name, contact_title, phone_number, email)
VALUES (
    '11',
    '111',
    'John Smith',
    'Sales Manager',
    '554567',
    'john@email.com'
);

-- for fourth system
-- 1
SELECT
    p.product_name AS product_name,
    p.category AS category,
    pv.pk_variant_id AS variant_sku,
    pv.variant_unit AS unit,
    pv.variant_unit_price AS unit_price
FROM
    t_products p
JOIN
    t_product_variants pv ON p.pk_product_id = pv.pk_product_id;

-- 2
SELECT
    p.product_name,
```

```
211     mag_product.pk_variant_id,
212     p.category,
213     mag_product.shelf_quantity AS current_shelf_quantity,
214     AVG(mag_product.shelf_quantity) OVER(PARTITION BY p.category) AS
        avg_shelf_quantity,
215     mag_product.warehouse_quantity AS warehouse_quantity,
216     CASE
217         WHEN mag_product.shelf_quantity < AVG(mag_product.shelf_quantity
        ) OVER(PARTITION BY p.category) THEN 'Yes'
218         ELSE 'No'
219     END AS whether_to_update_shelf
220 FROM
221     t_products p
222 JOIN
223     (SELECT pk_product_id, pk_variant_id, shelf_quantity,
        warehouse_quantity FROM t_product_variants) mag_product ON p.
        pk_product_id = mag_product.pk_product_id
224 ORDER BY
225     CASE WHEN whether_to_update_shelf = 'Yes' THEN 0 ELSE 1 END,
226     p.category,
227     p.product_name;
228
229 -- 3
230 -- Insert new product into `t_products`
231 INSERT INTO t_products (pk_product_id, product_name, category,
        description)
232 VALUES ('6666', 'New product name', 'New product category', 'New product
         description');
233 -- Get the newly inserted product's ID
234
235 -- For each new product variant, you need to repeat this INSERT
        statement, each time inserting a variant
236 INSERT INTO t_product_variants (pk_product_id, pk_variant_id,
        variant_name, unit, unit_price, description, warehouse_quantity,
        shelf_quantity)
237 VALUES ('6666', '1', 'Variant name', 'Unit', '3', 'awesome', '4', '5');
238
239 -- Delete existing stored procedure
240 DROP PROCEDURE IF EXISTS InsertSupplierAndContact;
241
242 -- Create new stored procedure
243 DELIMITER //
244 CREATE PROCEDURE InsertSupplierAndContact()
245 BEGIN
246     -- Start transaction
247     START TRANSACTION;
248
249     -- Attempt to insert into t_suppliers table
250     INSERT INTO t_suppliers (pk_supplier_id, company_name,
```

```
    company_address, company_website)
    SELECT '22', 'Haha', 'Awesome', 'Takeoff'
    WHERE NOT EXISTS (
        SELECT 1 FROM t_suppliers
        WHERE pk_supplier_id = '22'
        AND company_name = 'Haha'
        AND company_address = 'Awesome'
        AND company_website = 'Takeoff'
    );

    -- Check if any rows were inserted
    IF ROW_COUNT() > 0 THEN
        -- Insert into t_supplier_contacts
        INSERT INTO t_supplier_contacts (pk_contact_id, fk_supplier_id,
    contact_name, contact_title, phone_number, email)
        VALUES ('3333', '22', 'Hey', 'Ha', '123', 'sss');
    END IF;
    -- Directly insert into t_supply_records table
    INSERT INTO t_supply_records (pk_supply_record_id,
    fk_supply_product_id, fk_supply_variant_id, fk_supplier_id,
    supply_date, supply_quantity, total_price, pay_term)
    VALUES ('2222', '6666', '1', '22', '2023-04-01', '33', '33', 'Net 30
    days');

    -- Commit transaction
    COMMIT;
END //
DELIMITER ;

-- Now you can call the newly created stored procedure
CALL InsertSupplierAndContact();

-- 4
START TRANSACTION;

-- Assume variables have been set to specific values
-- management_date, management_time, responsible_employee_id, product_id
    , variant_id, warehouse_change_quantity, shelf_change_quantity
-- Please replace them with specific values or parameters

-- Insert a row into t_product_management_records table
INSERT INTO t_product_management_records
(management_date, management_time, fk_responsible_employee_id,
    fk_product_id, fk_variant_id, warehouse_change_quantity,
    shelf_change_quantity)
VALUES
(CURDATE(), CURTIME(), @responsible_employee_id, @product_id,
    @variant_id, @warehouse_change_quantity, @shelf_change_quantity);

```

```
-- Update t_product_variants table
-- Suppose you are updating the warehouse inventory quantity
UPDATE t_product_variants
SET warehouse_quantity = warehouse_quantity + @warehouse_change_quantity
WHERE pk_product_id = @product_id AND pk_variant_id = @variant_id;

-- If shelf inventory also needs to be updated, you can add this line
UPDATE t_product_variants
SET shelf_quantity = shelf_quantity + @shelf_change_quantity
WHERE pk_product_id = @product_id AND pk_variant_id = @variant_id;

-- 5
-- Assume variable @product_id has been set to a specific product ID
START TRANSACTION;

-- Update description in t_products table
UPDATE t_products
SET description = 'sold out'
WHERE pk_product_id = @product_id;

-- Update warehouse_quantity and shelf_quantity in t_product_variants
   table
UPDATE t_product_variants
SET warehouse_quantity = 0,
    shelf_quantity = 0
WHERE pk_product_id = @product_id;
```

Listing 2: SQL Queries for the Project

```
<template>
    <el-card>
        <div slot="header">
            <span>{{ title }}</span>
        </div>
        <el-form ref="insertForm" :model="newRow" size="mini" :inline="
   inline">
            <el-form-item v-for="column in columns" :label="column.label
   " :key="column.key">
                <el-input v-model="newRow[column.key]"></el-input>
            </el-form-item>
            <el-form-item>
                <el-button :type="button_type" @click="insertRow">{{
   button_text }}</el-button>
            </el-form-item>
        </el-form>
    </el-card>
</template>

<script>
import apiClient from '@/services/apiClient';
```

```
19
20 export default {
21     data() {
22         return {
23             newRow: {}
24         };
25     },
26     methods: {
27         insertRow() {
28             if (!this.is_post) {
29                 apiClient.patch(this.url, this.newRow)
30                     .then(() => {
31                         this.$message({
32                             message: this.message_success,
33                             type: 'success'
34                         });
35                         this.$emit('inserted');
36                     })
37                     .catch(() => {
38                         this.$message({
39                             message: this.message_error,
40                             type: 'error'
41                         });
42                     });
43                 return;
44             }
45             apiClient.post(this.url, this.newRow)
46                 .then(() => {
47                     this.$message({
48                         message: this.message_success,
49                         type: 'success'
50                     });
51                     this.$emit('inserted');
52                 })
53                 .catch(() => {
54                     this.$message({
55                         message: this.message_error,
56                         type: 'error'
57                     });
58                 });
59         }
60     },
61     props: {
62         title: {
63             type: String,
64             default: 'Default Title'
65         },
66         url: String,
67         columns: Array,
```

```
        inline: {
            type: Boolean,
            default: false
        },
        button_text: {
            type: String,
            default: 'Insert New Row'
        },
        message_success: {
            type: String,
            default: 'Row inserted successfully'
        },
        message_error: {
            type: String,
            default: 'Failed to insert row'
        },
        button_type: {
            type: String,
            default: 'primary'
        },
        is_post: {
            type: Boolean,
            default: true
        }
    },
    created() {
        // Initialize newRow with properties based on columns
        this.columns.forEach(column => {
            this.$set(this.newRow, column.key, '');
        });
    }
};
</script>

<style scoped>
.el-card {
    margin-top: 15px;
    margin-bottom: 15px;
    margin-left: 15px;
    margin-right: 15px;
}
</style>
```

Listing 3: Frontend Form Component

```
<template>
    <el-card style="margin-top: 20px; height: auto;">
        <h3 style="margin: 0 0 20px 0; padding: 10px;">{{ title }}</h3>
        <div v-if="showCharts">
            <div ref="barchart" style="width: 50%; height: 500px;
```

```
    display: inline-block;"></div>
6           <div ref="piechart" style="width: 50%; height: 500px;
    display: inline-block;"></div>
7       </div>
8       <div style="height: 360px; overflow-y: auto;">
9           <el-table :data="tableData" style="width: 100%">
10              <el-table-column v-for="key in columns" :key="key" :prop
    ="key"
11                  :label="key.replace(/_/g, ' ').toUpperCase()">
12              </el-table-column>
13              <el-table-column v-if="allowDelete" label="Actions">
14                  <template v-slot="scope">
15                      <el-button size="mini" v-if="allowDelete" type="
    danger"
16                          @click="deleteRow(scope.$index)">Delete</el-
    button>
17                  </template>
18              </el-table-column>
19          </el-table>
20      </div>
21   </el-card>
22 </template>
23
24 <script>
25 import apiClient from '@/services/apiClient';
26 import * as echarts from 'echarts';
27
28 export default {
29     data() {
30         return {
31             tableData: [],
32             refreshInterval: null
33         };
34     },
35     props: {
36         title: {
37             type: String,
38             default: 'Default Title'
39         },
40         url: String,
41         allowDelete: {
42             type: Boolean,
43             default: false
44         },
45         deleteKey: String,
46         deleteUrl: String,
47         showCharts: {
48             type: Boolean,
49             default: false
```

```
        },
        barLabelColumn: {
            type: String,
            default: ''
        },
        barDataColumns: {
            type: Array,
            default: () => ['', '']
        },
        pieLabelColumn: {
            type: String,
            default: ''
        },
        pieQuantityColumn: {
            type: String,
            default: ''
        }
    },
    computed: {
        columns() {
            return this.tableData.length > 0 ? Object.keys(this.
    tableData[0]) : [];
        }
    },
    mounted() {
        this.fetchData();
        this.setupRefresh();
        // sleep for 1 second to wait for the data to be fetched
        setTimeout(() => {
            if (this.showCharts) {
                this.initCharts();
            }
        }, 1000);
        if (this.showCharts) {
            this.initCharts();
        }
    },
    beforeDestroy() {
        if (this.refreshInterval) {
            clearInterval(this.refreshInterval);
        }
    },
    methods: {
        fetchData() {
            apiClient.get(this.url)
                .then(response => {
                    // turncate if too long
                    this.tableData = response.data.slice(0, 500);
                })
```

```
98                .catch(error => {
99                    console.error('Error fetching data:', error);
100               });
101       },
102       setupRefresh() {
103           this.refreshInterval = setInterval(this.fetchData, 100000);
    // Adjust for better effects
104       },
105       deleteRow(index) {
106           const id = this.tableData[index][this.deleteKey];
107           // if deleteKey is not in the tableData, raise an error
108           if (!id) {
109               console.error('deleteKey not found in tableData:', this.
    deleteKey);
110               return;
111           }
112           console.log('Deleting row with id:', id);
113           apiClient.post(this.deleteUrl, { [this.deleteKey]: id })
114               .then(() => {
115                   console.log('Row deleted successfully');
116               })
117               .catch(error => {
118                   console.error('Error deleting row:', error);
119               });
120           // remove the row from the tableData
121           this.tableData.splice(index, 1);
122       },
123       initCharts() {
124           var barOptions = {
125               xAxis: {
126                   data: this.tableData.map(item => item[this.
    barLabelColumn]),
127                   axisLabel: {
128                       // x-axis label
129                       show: false
130                   }
131               },
132               yAxis: {},
133               series: this.barDataColumns.map(column => ({
134                   type: 'bar',
135                   data: this.tableData.map(item => item[column])
136               })),
137               tooltip: {
138                   trigger: 'axis',
139                   axisPointer: {
140                       type: 'shadow'
141                   }
142               }
143
```

```
            };
            console.log('barOptions:', barOptions);
            var pieOptions = {
                legend: {
                    orient: 'vertical',
                    x: 'left',
                    data: this.tableData.map(item => item[this.
    pieLabelColumn])
                },
                series: [
                    {
                        type: 'pie',
                        radius: ['50%', '70%'],
                        avoidLabelOverlap: false,
                        label: {
                            show: false,
                            position: 'center'
                        },
                        labelLine: {
                            show: false
                        },
                        emphasis: {
                            label: {
                                show: true,
                                fontSize: '30',
                                fontWeight: 'bold'
                            }
                        },
                        data: this.tableData.map(item => ({
                            value: item[this.pieQuantityColumn],
                            name: item[this.pieLabelColumn]
                        }))
                    }
                ]
            };
            var barchart = echarts.init(this.$refs.barchart);
            barchart.setOption(barOptions);
            var piechart = echarts.init(this.$refs.piechart);
            piechart.setOption(pieOptions);
        }
    }
}
</script>
```

Listing 4: Frontend Table Component

```
-- MySQL dump 10.13  Distrib 8.0.35, for Win64 (x86_64)
--
-- Host: localhost    Database: comprehensive_supermarket
-- ------------------------------------------------------
```

```
-- Server version 8.0.35
CREATE DATABASE IF NOT EXISTS comprehensive_supermarket;
SHOW DATABASES;
USE comprehensive_supermarket;

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
    FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
    */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `t_customers`
--


DROP TABLE IF EXISTS `t_customers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_customers` (
  `pk_customer_id` int NOT NULL,
  `customer_name` varchar(255) DEFAULT NULL,
  `phone_number` int DEFAULT NULL,
  `account_balance` float DEFAULT NULL,
  PRIMARY KEY (`pk_customer_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_customers`
--

LOCK TABLES `t_customers` WRITE;
/*!40000 ALTER TABLE `t_customers` DISABLE KEYS */;
INSERT INTO `t_customers` VALUES (1,'John Doe',1234567890,250.5),(2,'
    Jane Smith',1234567891,150.75),(3,'Bob Johnson',1234567892,300),(4,'
    Alice Williams',1234567893,275.25),(5,'Chris Brown',1234567894,125)
    ,(6,'Diana Clark',1234567895,180.45),(7,'Evan Davis'
    ,1234567896,220.3),(8,'Fiona Evans',1234567897,205.6),(9,'Gary Harris
    ',1234567898,190.85),(10,'Helen Jackson',1234567899,260.7),(11,'Ian
    King',1234567900,140.55),(12,'Jessica Lee',1234567901,230.2),(13,'
    Kyle Martin',1234567902,210.9),(14,'Laura Nelson',1234567903,240)
```

```
,(15,'Mike O\'Neil',1234567904,200.15),(16,'Nina Perez'
,1234567905,175.8),(17,'Oscar Quinn',1234567906,195.45),(18,'Patricia
 Robinson',1234567907,215.1),(19,'Quinn Stevens',1234567908,235.75)
,(20,'Rachel Taylor',1234567909,255.4),(21,'Steven Underwood'
,1234567910,275.05),(22,'Tina Vincent',1234567911,294.7),(23,'Ursula
Wilson',1234567912,314.35),(24,'Victor Xander',1234567913,334),(25,'
Wendy Young',1234567914,353.65),(26,'Xavier Zane',1234567915,373.3)
,(27,'Yolanda Adams',1234567916,392.95),(28,'Zachary Brooks'
,1234567917,412.6),(29,'Amanda Carter',1234567918,432.25),(30,'Brian
Daniels',1234567919,451.9),(31,'Caroline Edwards',1234567920,471.55)
,(32,'Derek Franklin',1234567921,491.2),(33,'Eleanor Green'
,1234567922,510.85),(34,'Franklin Hopper',1234567923,530.5),(35,'
Georgia Ingram',1234567924,550.15),(36,'Harold Jenkins'
,1234567925,569.8),(37,'Iris Kent',1234567926,589.45),(38,'Justin
Lopez',1234567927,609.1),(39,'Karen Moore',1234567928,628.75),(40,'
Louis Norton',1234567929,648.4),(41,'Margaret O\'Connor'
,1234567930,668.05),(42,'Nathaniel Peters',1234567931,687.7),(43,'
Olivia Queen',1234567932,707.35),(44,'Peter Russell',1234567933,727)
,(45,'Quincy Simmons',1234567934,746.65),(46,'Renee Thomas'
,1234567935,766.3),(47,'Simon Upton',1234567936,785.95),(48,'Teresa
Vaughn',1234567937,805.6),(49,'Ulysses Wallace',1234567938,825.25)
,(50,'Vanessa Young',1234567939,844.9),(51,'Alexa Ray'
,1276543201,305.2),(52,'Benjamin Knight',1276543202,150.75),(53,'
Charlotte Lane',1276543203,210.55),(54,'Dexter Morgan'
,1276543204,500),(55,'Evelyn Stone',1276543205,275.45),(56,'Frank
Ocean',1276543206,325.3),(57,'Grace Hart',1276543207,205.6),(58,'
Henry Ford',1276543208,190.85),(59,'Isabella King',1276543209,260.7)
,(60,'Jack Ryan',1276543210,340.55),(61,'Kate Marsh'
,1276543211,230.2),(62,'Liam Neeson',1276543212,410.9),(63,'Mia
Wallace',1276543213,240),(64,'Noah Flynn',1276543214,200.15),(65,'
Olivia Pope',1276543215,375.8),(66,'Peter Parker',1276543216,195.45)
,(67,'Quinn Fabray',1276543217,215.1),(68,'Rachel Green'
,1276543218,235.75),(69,'Sam Winchester',1276543219,255.4),(70,'Tina
Cohen',1276543220,275.05),(71,'Ursula Monroe',1276543221,294.7),(72,'
Vincent Vega',1276543222,314.35),(73,'Walter White',1276543223,334)
,(74,'Xena Warrior',1276543224,353.65),(75,'Yvonne Strahovski'
,1276543225,373.3),(76,'Zachary Levi',1276543226,392.95),(77,'Adam
West',1276543227,412.6),(78,'Betty Cooper',1276543228,432.25),(79,'
Clark Kent',1276543229,451.9),(80,'Diana Prince',1276543230,471.55)
,(81,'Ethan Hunt',1276543231,491.2),(82,'Fiona Gallagher'
,1276543232,510.85),(83,'George Bluth',1276543233,530.5),(84,'Harley
Quinn',1276543234,550.15),(85,'Ivy Dickens',1276543235,569.8),(86,'
Joey Tribbiani',1276543236,589.45),(87,'Kara Danvers'
,1276543237,609.1),(88,'Lucifer Morningstar',1276543238,628.75),(89,'
Mona Vanderwaal',1276543239,648.4),(90,'Nancy Wheeler'
,1276543240,668.05),(91,'Oscar Bluth',1276543241,687.7),(92,'Phoebe
Buffay',1276543242,707.35),(93,'Quentin Coldwater',1276543243,727)
,(94,'Rory Gilmore',1276543244,746.65),(95,'Selina Meyer'
,1276543245,766.3),(96,'Tony Stark',1276543246,785.95),(97,'Uma
```

```
    Thurman',1276543247,805.6),(98,'Vanessa Ives',1276543248,825.25),(99,
    'Wade Wilson',1276543249,844.9),(100,'Xander Harris'
    ,1276543250,864.55);
/*!40000 ALTER TABLE `t_customers` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_employees`
--

DROP TABLE IF EXISTS `t_employees`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_employees` (
  `pk_employee_id` int NOT NULL,
  `job_position` varchar(255) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  `phone_number` int DEFAULT NULL,
  `work_schedules` text,
  PRIMARY KEY (`pk_employee_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_employees`
--

LOCK TABLES `t_employees` WRITE;
/*!40000 ALTER TABLE `t_employees` DISABLE KEYS */;
INSERT INTO `t_employees` VALUES (1,'Cashier',1800,1234567890,'Mon-Fri
    09:00-17:00'),(2,'Cashier',1850,1234567891,'Mon-Fri 10:00-18:00'),(3,
    'Cashier',1900,1234567892,'Mon-Fri 11:00-19:00'),(4,'Cashier'
    ,1800,1234567893,'Mon-Fri 12:00-20:00'),(5,'Cashier',1750,1234567894,
    'Mon-Fri 13:00-21:00'),(6,'Cashier',1950,1234567895,'Tue-Sat
    09:00-17:00'),(7,'Cashier',1825,1234567896,'Tue-Sat 10:00-18:00'),(8,
    'Cashier',1875,1234567897,'Tue-Sat 11:00-19:00'),(9,'Cashier'
    ,1925,1234567898,'Tue-Sat 12:00-20:00'),(10,'Cashier'
    ,1830,1234567899,'Tue-Sat 13:00-21:00'),(11,'Cashier'
    ,1930,1234567800,'Wed-Sun 09:00-17:00'),(12,'Cashier'
    ,1780,1234567801,'Wed-Sun 10:00-18:00'),(13,'Cashier'
    ,1880,1234567802,'Wed-Sun 11:00-19:00'),(14,'Cashier'
    ,1750,1234567803,'Wed-Sun 12:00-20:00'),(15,'Cashier'
    ,1900,1234567804,'Wed-Sun 13:00-21:00'),(16,'Cashier'
    ,1850,1234567805,'Thu-Mon 09:00-17:00'),(17,'Cashier'
    ,1800,1234567806,'Thu-Mon 10:00-18:00'),(18,'Cashier'
    ,1850,1234567807,'Thu-Mon 11:00-19:00'),(19,'Cashier'
    ,1800,1234567808,'Thu-Mon 12:00-20:00'),(20,'Cashier'
    ,1780,1234567809,'Thu-Mon 13:00-21:00'),(21,'Store Organizer'
    ,1000,1234567810,'Mon-Fri 08:00-16:00'),(22,'Store Organizer'
```

```
    ,1100,1234567811,'Mon-Fri 09:00-17:00'),(23,'Store Organizer'
    ,1200,1234567812,'Tue-Sat 08:00-16:00'),(24,'Store Organizer'
    ,1050,1234567813,'Tue-Sat 09:00-17:00'),(25,'Store Organizer'
    ,1150,1234567814,'Wed-Sun 08:00-16:00'),(26,'Store Organizer'
    ,1250,1234567815,'Wed-Sun 09:00-17:00'),(27,'Store Organizer'
    ,1000,1234567816,'Thu-Mon 08:00-16:00'),(28,'Store Organizer'
    ,1100,1234567817,'Thu-Mon 09:00-17:00'),(29,'Store Organizer'
    ,1200,1234567818,'Fri-Tue 08:00-16:00'),(30,'Store Organizer'
    ,1050,1234567819,'Fri-Tue 09:00-17:00'),(31,'Security Guard'
    ,1200,1234567820,'Mon-Fri 06:00-14:00'),(32,'Security Guard'
    ,1300,1234567821,'Mon-Fri 14:00-22:00'),(33,'Security Guard'
    ,1250,1234567822,'Tue-Sat 06:00-14:00'),(34,'Security Guard'
    ,1350,1234567823,'Tue-Sat 14:00-22:00'),(35,'Security Guard'
    ,1400,1234567824,'Wed-Sun 06:00-14:00'),(36,'Floor Supervisor'
    ,2500,1234567825,'Mon-Fri 08:00-16:00'),(37,'Floor Supervisor'
    ,2600,1234567826,'Mon-Fri 12:00-20:00'),(38,'Floor Supervisor'
    ,2550,1234567827,'Tue-Sat 08:00-16:00'),(39,'Floor Supervisor'
    ,2650,1234567828,'Tue-Sat 12:00-20:00'),(40,'Floor Supervisor'
    ,2700,1234567829,'Wed-Sun 08:00-16:00'),(41,'Floor Supervisor'
    ,2750,1234567830,'Wed-Sun 12:00-20:00'),(42,'Bakery Clerk'
    ,1900,1234567831,'Mon-Fri 05:00-13:00'),(43,'Butcher'
    ,2000,1234567832,'Mon-Fri 06:00-14:00'),(44,'Produce Associate'
    ,1850,1234567833,'Mon-Fri 07:00-15:00'),(45,'Seafood Specialist'
    ,1950,1234567834,'Mon-Fri 06:00-14:00'),(46,'Grocery Stocker'
    ,1800,1234567835,'Tue-Sat 07:00-15:00'),(47,'Dairy Associate'
    ,1850,1234567836,'Tue-Sat 06:00-14:00'),(48,'Delicatessen Clerk'
    ,1900,1234567837,'Tue-Sat 07:00-15:00'),(49,'Floral Designer'
    ,1950,1234567838,'Wed-Sun 09:00-17:00'),(50,'Pharmacy Technician'
    ,2200,1234567839,'Wed-Sun 08:00-16:00'),(51,'Checkout Supervisor'
    ,2100,1234567840,'Thu-Mon 10:00-18:00'),(52,'Wine Steward'
    ,2050,1234567841,'Thu-Mon 11:00-19:00'),(53,'Bakery Manager'
    ,2500,1234567842,'Thu-Mon 05:00-13:00'),(54,'Meat Department Manager'
    ,2600,1234567843,'Fri-Tue 06:00-14:00'),(55,'Produce Manager'
    ,2550,1234567844,'Fri-Tue 07:00-15:00'),(56,'Seafood Manager'
    ,2650,1234567845,'Mon-Fri 06:00-14:00'),(57,'Grocery Manager'
    ,2700,1234567846,'Mon-Fri 07:00-15:00'),(58,'Dairy Manager'
    ,2450,1234567847,'Tue-Sat 06:00-14:00'),(59,'Delicatessen Manager'
    ,2550,1234567848,'Tue-Sat 07:00-15:00'),(60,'Floral Department
   Manager',2650,1234567849,'Wed-Sun 09:00-17:00');
/*!40000 ALTER TABLE `t_employees` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_expense_reports`
--

DROP TABLE IF EXISTS `t_expense_reports`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
```

```
CREATE TABLE `t_expense_reports` (
  `pk_expense_report_id` int NOT NULL,
  `report_type` varchar(255) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `create_time` datetime DEFAULT NULL,
  `supply_expense` float DEFAULT NULL,
  `employee_expense` float DEFAULT NULL,
  `other_expense` float DEFAULT NULL,
  `total_expense` float DEFAULT NULL,
  PRIMARY KEY (`pk_expense_report_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_expense_reports`
--

LOCK TABLES `t_expense_reports` WRITE;
/*!40000 ALTER TABLE `t_expense_reports` DISABLE KEYS */;
INSERT INTO `t_expense_reports` VALUES (1,'Monthly','2023-02-01','
    2023-02-28','2023-03-01 10:00:00',5000,2000,800,7800),(2,'Quarterly',
    '2023-01-01','2023-03-31','2023-04-01 10:00:00'
    ,15000,6000,2400,21600),(3,'Annual','2022-01-01','2022-12-31','
    2023-01-01 10:00:00',60000,24000,9600,93600),(4,'Monthly','2023-03-01
    ','2023-03-31','2023-04-01 10:00:00',5200,2100,850,8150),(5,'Weekly',
    '2023-03-01','2023-03-07','2023-03-08 10:00:00',1200,500,200,1900)
    ,(6,'Monthly','2023-01-01','2023-01-31','2023-02-01 10:00:00'
    ,4800,1900,770,7470),(7,'Quarterly','2023-04-01','2023-06-30','
    2023-07-01 10:00:00',15500,6200,2500,24200),(8,'Annual','2023-01-01',
    '2023-12-31','2024-01-01 10:00:00',61000,25000,10000,96000),(9,'
    Weekly','2023-03-08','2023-03-14','2023-03-15 10:00:00'
    ,1300,550,220,2070),(10,'Monthly','2023-04-01','2023-04-30','
    2023-05-01 10:00:00',5300,2200,880,8380);
/*!40000 ALTER TABLE `t_expense_reports` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_inventory_reports`
--

DROP TABLE IF EXISTS `t_inventory_reports`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_inventory_reports` (
  `pk_inventory_report_id` int NOT NULL,
  `report_type` varchar(255) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
```

```
  `create_time` datetime DEFAULT NULL,
  PRIMARY KEY (`pk_inventory_report_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_inventory_reports`
--

LOCK TABLES `t_inventory_reports` WRITE;
/*!40000 ALTER TABLE `t_inventory_reports` DISABLE KEYS */;
INSERT INTO `t_inventory_reports` VALUES (1,'Daily','2023-03-01','
    2023-03-01','2023-03-02 08:00:00'),(2,'Weekly','2023-02-24','
    2023-03-02','2023-03-03 08:00:00'),(3,'Monthly','2023-02-01','
    2023-02-28','2023-03-01 08:00:00'),(4,'Quarterly','2023-01-01','
    2023-03-31','2023-04-01 08:00:00'),(5,'Yearly','2022-01-01','
    2022-12-31','2023-01-01 08:00:00'),(6,'Daily','2023-03-02','
    2023-03-02','2023-03-03 08:00:00'),(7,'Weekly','2023-03-03','
    2023-03-09','2023-03-10 08:00:00'),(8,'Monthly','2023-03-01','
    2023-03-31','2023-04-01 08:00:00'),(9,'Quarterly','2023-04-01','
    2023-06-30','2023-07-01 08:00:00'),(10,'Yearly','2023-01-01','
    2023-12-31','2024-01-01 08:00:00');
/*!40000 ALTER TABLE `t_inventory_reports` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_product_management_records`
--

DROP TABLE IF EXISTS `t_product_management_records`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_product_management_records` (
  `pk_management_record_id` int NOT NULL,
  `management_date` date DEFAULT NULL,
  `management_time` time DEFAULT NULL,
  `fk_responsible_employee_id` int DEFAULT NULL,
  `fk_product_id` int DEFAULT NULL,
  `fk_variant_id` int DEFAULT NULL,
  `warehouse_change_quantity` float DEFAULT NULL,
  `shelf_change_quantity` float DEFAULT NULL,
  PRIMARY KEY (`pk_management_record_id`),
  KEY `fk_responsible_employee_id` (`fk_responsible_employee_id`),
  KEY `fk_product_id` (`fk_product_id`,`fk_variant_id`),
  CONSTRAINT `product_management_records_ibfk_1` FOREIGN KEY (`
    fk_responsible_employee_id`) REFERENCES `t_employees` (`
    pk_employee_id`),
  CONSTRAINT `product_management_records_ibfk_2` FOREIGN KEY (`
    fk_product_id`, `fk_variant_id`) REFERENCES `t_product_variants` (`
```

```
   pk_product_id`, `pk_variant_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_product_management_records`
--

LOCK TABLES `t_product_management_records` WRITE;
/*!40000 ALTER TABLE `t_product_management_records` DISABLE KEYS */;
INSERT INTO `t_product_management_records` VALUES (1,'2023-01-10','
   08:00:00',21,1,1,-5,5),(2,'2023-01-10','09:00:00',22,1,2,-10,10),(3,'
   2023-01-11','10:30:00',23,2,1,-8,8),(4,'2023-01-11','11:00:00'
   ,24,2,2,-6,6),(5,'2023-01-12','08:30:00',25,3,1,-4,4),(6,'2023-01-12'
   ,'09:30:00',26,3,2,-2,2),(7,'2023-01-13','10:15:00',27,4,1,-7,7),(8,'
   2023-01-13','10:45:00',28,4,2,-3,3),(9,'2023-01-14','08:20:00'
   ,29,5,1,-6,6),(10,'2023-01-14','09:10:00',30,5,2,-8,8),(11,'
   2023-01-15','11:00:00',21,6,1,-5,5),(12,'2023-01-15','12:00:00'
   ,22,6,2,-10,10),(13,'2023-01-16','08:00:00',23,7,1,-3,3),(14,'
   2023-01-16','08:30:00',24,7,2,-4,4),(15,'2023-01-17','09:00:00'
   ,25,8,1,-7,7),(16,'2023-01-17','10:00:00',26,8,2,-2,2),(17,'
   2023-01-18','08:45:00',27,9,1,-1,1),(18,'2023-01-18','09:50:00'
   ,28,9,2,-5,5),(19,'2023-01-19','08:30:00',29,10,1,-8,8),(20,'
   2023-01-19','09:30:00',30,10,2,-7,7),(21,'2023-01-20','10:15:00'
   ,21,11,1,-6,6),(22,'2023-01-20','11:15:00',22,11,2,-3,3),(23,'
   2023-01-21','08:00:00',23,12,1,-4,4),(24,'2023-01-21','08:45:00'
   ,24,12,2,-5,5),(25,'2023-01-22','09:30:00',25,13,1,-2,2),(26,'
   2023-01-22','10:30:00',26,13,2,-3,3),(27,'2023-01-23','08:20:00'
   ,27,14,1,-4,4),(28,'2023-01-23','09:40:00',28,14,2,-2,2),(29,'
   2023-01-24','10:30:00',29,15,1,-5,5),(30,'2023-01-24','11:00:00'
   ,30,15,2,-3,3),(31,'2023-01-25','08:15:00',21,16,1,-7,7),(32,'
   2023-01-25','09:45:00',22,16,2,-1,1),(33,'2023-01-26','10:10:00'
   ,23,17,1,-8,8),(34,'2023-01-26','11:30:00',24,17,2,-4,4),(35,'
   2023-01-27','08:25:00',25,18,1,-3,3),(36,'2023-01-27','09:55:00'
   ,26,18,2,-6,6),(37,'2023-01-28','10:40:00',27,19,1,-5,5),(38,'
   2023-01-28','11:20:00',28,19,2,-2,2),(39,'2023-01-29','08:30:00'
   ,29,20,1,-4,4),(40,'2023-01-29','09:30:00',30,20,2,-7,7),(41,'
   2023-01-30','10:15:00',21,21,1,-8,8),(42,'2023-01-30','11:15:00'
   ,22,21,2,-1,1),(43,'2023-01-31','08:00:00',23,22,1,-5,5),(44,'
   2023-01-31','09:00:00',24,22,2,-3,3),(45,'2023-02-01','10:30:00'
   ,25,23,1,-6,6),(46,'2023-02-01','11:00:00',26,23,2,-4,4),(47,'
   2023-02-02','08:30:00',27,24,1,-2,2),(48,'2023-02-02','09:30:00'
   ,28,24,2,-5,5),(49,'2023-02-03','10:15:00',29,25,1,-7,7),(50,'
   2023-02-03','11:15:00',30,25,2,-3,3),(51,'2023-02-04','08:00:00'
   ,21,26,1,-1,1),(52,'2023-02-04','09:00:00',22,26,2,-6,6),(53,'
   2023-02-05','10:30:00',23,27,1,-4,4),(54,'2023-02-05','11:00:00'
   ,24,27,2,-8,8),(55,'2023-02-06','08:30:00',25,28,1,-2,2),(56,'
   2023-02-06','09:30:00',26,28,2,-7,7),(57,'2023-02-07','10:15:00'
   ,27,29,1,-3,3),(58,'2023-02-07','11:15:00',28,29,2,-5,5),(59,'
```

```
2023-02-08','08:00:00',29,30,1,-4,4),(60,'2023-02-08','09:00:00'
,30,30,2,-6,6),(61,'2023-04-01','08:00:00',21,16,1,-10,10),(62,'
2023-04-01','08:15:00',22,16,2,5,-5),(63,'2023-04-01','08:30:00'
,23,17,1,-15,15),(64,'2023-04-01','08:45:00',24,17,2,20,-20),(65,'
2023-04-01','09:00:00',25,18,1,-5,5),(66,'2023-04-01','09:15:00'
,26,18,2,10,-10),(67,'2023-04-01','09:30:00',27,19,1,-20,20),(68,'
2023-04-01','09:45:00',28,19,2,15,-15),(69,'2023-04-01','10:00:00'
,29,20,1,-25,25),(70,'2023-04-01','10:15:00',30,20,2,30,-30),(71,'
2023-04-01','10:30:00',21,21,1,-5,5),(72,'2023-04-01','10:45:00'
,22,21,2,10,-10),(73,'2023-04-01','11:00:00',23,22,1,-15,15),(74,'
2023-04-01','11:15:00',24,22,2,20,-20),(75,'2023-04-01','11:30:00'
,25,23,1,-10,10),(76,'2023-04-01','11:45:00',26,23,2,5,-5),(77,'
2023-04-01','12:00:00',27,16,1,-30,30),(78,'2023-04-01','12:15:00'
,28,17,1,25,-25),(79,'2023-04-01','12:30:00',29,18,1,-5,5),(80,'
2023-04-01','12:45:00',30,19,1,15,-15),(81,'2023-04-01','13:00:00'
,21,20,1,-20,20),(82,'2023-04-01','13:15:00',22,21,1,10,-10),(83,'
2023-04-01','13:30:00',23,22,1,-15,15),(84,'2023-04-01','13:45:00'
,24,23,1,20,-20),(85,'2023-04-01','14:00:00',25,16,2,-10,10),(86,'
2023-04-01','14:15:00',26,17,2,5,-5),(87,'2023-04-01','14:30:00'
,27,18,2,-25,25),(88,'2023-04-01','14:45:00',28,19,2,30,-30),(89,'
2023-04-01','15:00:00',29,20,2,-5,5),(90,'2023-04-01','15:15:00'
,30,21,2,15,-15),(91,'2023-04-01','15:30:00',21,22,2,-20,20),(92,'
2023-04-01','15:45:00',22,23,2,10,-10),(93,'2023-04-01','08:00:00'
,21,24,1,-5,5),(94,'2023-04-01','09:00:00',22,25,2,10,-10),(95,'
2023-04-01','10:00:00',23,26,1,-15,15),(96,'2023-04-01','11:00:00'
,24,27,2,20,-20),(97,'2023-04-01','12:00:00',25,28,1,-25,25),(98,'
2023-04-01','13:00:00',26,29,2,30,-30),(99,'2023-04-01','14:00:00'
,27,30,1,-35,35),(100,'2023-04-01','15:00:00',28,31,2,40,-40),(101,'
2023-04-02','08:00:00',29,32,1,-45,45),(102,'2023-04-02','09:00:00'
,30,24,2,50,-50),(103,'2023-04-02','10:00:00',21,25,1,-5,5),(104,'
2023-04-02','11:00:00',22,26,2,10,-10),(105,'2023-04-02','12:00:00'
,23,27,1,-15,15),(106,'2023-04-02','13:00:00',24,28,2,20,-20),(107,'
2023-04-02','14:00:00',25,29,1,-25,25),(108,'2023-04-02','15:00:00'
,26,30,2,30,-30),(109,'2023-04-03','08:00:00',27,31,1,-35,35),(110,'
2023-04-03','09:00:00',28,32,2,40,-40),(111,'2023-04-03','10:00:00'
,29,24,1,-45,45),(112,'2023-04-03','11:00:00',30,25,2,50,-50),(113,'
2023-04-03','12:00:00',21,26,1,-5,5),(114,'2023-04-03','13:00:00'
,22,27,2,10,-10),(115,'2023-04-03','14:00:00',23,28,1,-15,15),(116,'
2023-04-03','15:00:00',24,29,2,20,-20),(117,'2023-04-04','08:00:00'
,25,30,1,-25,25),(118,'2023-04-04','09:00:00',26,31,2,30,-30),(119,'
2023-04-04','10:00:00',27,32,1,-35,35),(120,'2023-04-04','11:00:00'
,28,24,2,40,-40),(121,'2023-04-04','12:00:00',29,25,1,-45,45),(122,'
2023-04-04','13:00:00',30,26,2,50,-50),(123,'2023-04-04','14:00:00'
,21,27,1,-5,5),(124,'2023-04-04','15:00:00',22,28,2,10,-10),(125,'
2023-04-01','09:00:00',21,33,1,-5,5),(126,'2023-04-01','09:15:00'
,22,33,2,10,-10),(127,'2023-04-01','09:30:00',23,34,1,-3,3),(128,'
2023-04-01','09:45:00',24,34,2,4,-4),(129,'2023-04-01','10:00:00'
,25,35,1,-7,7),(130,'2023-04-01','10:15:00',26,35,2,15,-15),(131,'
2023-04-01','10:30:00',27,36,1,-4,4),(132,'2023-04-01','10:45:00'
```

```
,28,36,2,5,-5),(133,'2023-04-01','11:00:00',29,37,1,-6,6),(134,'
2023-04-01','11:15:00',30,37,2,20,-20),(135,'2023-04-01','11:30:00'
,21,38,1,-5,5),(136,'2023-04-01','11:45:00',22,38,2,10,-10),(137,'
2023-04-01','12:00:00',23,39,1,-2,2),(138,'2023-04-01','12:15:00'
,24,39,2,3,-3),(139,'2023-04-01','12:30:00',25,40,1,-8,8),(140,'
2023-04-01','12:45:00',26,40,2,10,-10),(141,'2023-04-02','09:00:00'
,27,33,1,-10,10),(142,'2023-04-02','09:15:00',28,33,2,5,-5),(143,'
2023-04-02','09:30:00',29,34,1,-2,2),(144,'2023-04-02','09:45:00'
,30,34,2,7,-7),(145,'2023-04-02','10:00:00',21,35,1,-9,9),(146,'
2023-04-02','10:15:00',22,35,2,12,-12),(147,'2023-04-02','10:30:00'
,23,36,1,-3,3),(148,'2023-04-02','10:45:00',24,36,2,6,-6),(149,'
2023-04-02','11:00:00',25,37,1,-5,5),(150,'2023-04-02','11:15:00'
,26,37,2,15,-15),(151,'2023-04-02','11:30:00',27,38,1,-4,4),(152,'
2023-04-02','11:45:00',28,38,2,8,-8),(153,'2023-04-02','12:00:00'
,29,39,1,-7,7),(154,'2023-04-02','12:15:00',30,39,2,5,-5),(155,'
2023-04-02','12:30:00',21,40,1,-6,6),(156,'2023-04-02','12:45:00'
,22,40,2,11,-11),(157,'2023-03-01','09:00:00',21,41,1,-5,5),(158,'
2023-03-01','10:00:00',22,41,2,10,-10),(159,'2023-03-02','11:00:00'
,23,42,1,-15,15),(160,'2023-03-02','12:00:00',24,42,2,20,-20),(161,'
2023-03-03','13:00:00',25,43,1,-5,5),(162,'2023-03-03','14:00:00'
,26,43,2,10,-10),(163,'2023-03-04','15:00:00',27,44,1,-10,10),(164,'
2023-03-04','16:00:00',28,44,2,15,-15),(165,'2023-03-05','17:00:00'
,29,45,1,-20,20),(166,'2023-03-05','18:00:00',30,45,2,25,-25),(167,'
2023-03-06','08:00:00',21,46,1,-30,30),(168,'2023-03-06','09:00:00'
,22,46,2,35,-35),(169,'2023-03-07','10:00:00',23,47,1,-10,10),(170,'
2023-03-07','11:00:00',24,47,2,15,-15),(171,'2023-03-08','12:00:00'
,25,48,1,-5,5),(172,'2023-03-08','13:00:00',26,48,2,10,-10),(173,'
2023-03-09','14:00:00',27,49,1,-15,15),(174,'2023-03-09','15:00:00'
,28,49,2,20,-20),(175,'2023-03-10','16:00:00',29,50,1,-25,25),(176,'
2023-03-10','17:00:00',30,50,2,30,-30),(177,'2023-03-11','08:00:00'
,21,41,1,5,-5),(178,'2023-03-11','09:00:00',22,41,2,-10,10),(179,'
2023-03-12','10:00:00',23,42,1,15,-15),(180,'2023-03-12','11:00:00'
,24,42,2,-20,20),(181,'2023-03-13','12:00:00',25,43,1,5,-5),(182,'
2023-03-13','13:00:00',26,43,2,-10,10),(183,'2023-03-14','14:00:00'
,27,44,1,10,-10),(184,'2023-03-14','15:00:00',28,44,2,-15,15),(185,'
2023-03-15','16:00:00',29,45,1,20,-20),(186,'2023-03-15','17:00:00'
,30,45,2,-25,25),(187,'2023-03-16','08:00:00',21,46,1,30,-30),(188,'
2023-03-16','09:00:00',22,46,2,-35,35),(189,'2023-03-01','08:00:00'
,21,51,1,-10,10),(190,'2023-03-01','09:00:00',22,51,2,5,-5),(191,'
2023-03-01','10:00:00',23,52,1,-15,15),(192,'2023-03-01','11:00:00'
,24,52,2,20,-20),(193,'2023-03-01','12:00:00',25,53,1,-5,5),(194,'
2023-03-01','13:00:00',26,53,2,10,-10),(195,'2023-03-01','14:00:00'
,27,54,1,-20,20),(196,'2023-03-01','15:00:00',28,54,2,15,-15),(197,'
2023-03-01','16:00:00',29,55,1,-10,10),(198,'2023-03-02','08:00:00'
,30,55,2,5,-5),(199,'2023-03-02','09:00:00',21,56,1,-8,8),(200,'
2023-03-02','10:00:00',22,56,2,12,-12),(201,'2023-03-02','11:00:00'
,23,57,1,-7,7),(202,'2023-03-02','12:00:00',24,57,2,9,-9),(203,'
2023-03-02','13:00:00',25,58,1,-20,20),(204,'2023-03-02','14:00:00'
,26,58,2,15,-15),(205,'2023-03-02','15:00:00',27,59,1,-6,6),(206,'
```

```
    2023-03-02','16:00:00',28,59,2,4,-4),(207,'2023-03-03','08:00:00'
    ,29,60,1,-9,9),(208,'2023-03-03','09:00:00',30,60,2,11,-11),(209,'
    2023-03-03','10:00:00',21,51,1,-5,5),(210,'2023-03-03','11:00:00'
    ,22,51,2,3,-3),(211,'2023-03-03','12:00:00',23,52,1,-12,12),(212,'
    2023-03-03','13:00:00',24,52,2,18,-18),(213,'2023-03-03','14:00:00'
    ,25,53,1,-4,4),(214,'2023-03-03','15:00:00',26,53,2,6,-6),(215,'
    2023-03-03','16:00:00',27,54,1,-11,11),(216,'2023-03-04','08:00:00'
    ,28,54,2,7,-7),(217,'2023-03-04','09:00:00',29,55,1,-13,13),(218,'
    2023-03-04','10:00:00',30,55,2,14,-14),(219,'2023-03-04','11:00:00'
    ,21,56,1,-2,2),(220,'2023-03-04','12:00:00',22,56,2,8,-8);
/*!40000 ALTER TABLE `t_product_management_records` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_product_variants`
--

DROP TABLE IF EXISTS `t_product_variants`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_product_variants` (
  `pk_product_id` int NOT NULL,
  `pk_variant_id` int NOT NULL,
  `variant_name` varchar(255) DEFAULT NULL,
  `variant_unit` varchar(255) DEFAULT NULL,
  `variant_unit_price` float DEFAULT NULL,
  `variant_description` text,
  `warehouse_quantity` float DEFAULT NULL,
  `shelf_quantity` float DEFAULT NULL,
  PRIMARY KEY (`pk_product_id`,`pk_variant_id`),
  CONSTRAINT `t_product_variants_ibfk_1` FOREIGN KEY (`pk_product_id`)
   REFERENCES `t_products` (`pk_product_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_product_variants`
--

LOCK TABLES `t_product_variants` WRITE;
/*!40000 ALTER TABLE `t_product_variants` DISABLE KEYS */;
INSERT INTO `t_product_variants` VALUES (1,1,'Organic Red Apples (Bag of
    10)','Bag',5.99,'A bag of 10 crisp and sweet organic apples from
    local orchards.',100,20),(1,2,'Organic Red Apples (Single)','Each'
    ,0.69,'A single crisp and sweet organic apple from local orchards.'
    ,200,50),(2,1,'Whole Wheat Bread (Loaf)','Loaf',2.99,'Freshly baked
    loaf of bread made with 100% whole wheat flour.',80,30),(2,2,'Whole
    Wheat Bread (Sliced)','Pack',3.49,'Freshly baked bread made with 100%
    whole wheat flour, pre-sliced.',60,25),(3,1,'Atlantic Salmon (
```

```
Filleted)','Pound',8.99,'Fresh Atlantic salmon fillets, rich in Omega
-3 fatty acids.',50,10),(3,2,'Atlantic Salmon (Whole)','Each',17.99,'
A whole fresh Atlantic salmon, rich in Omega-3 fatty acids.',30,5)
,(4,1,'Angus Beef Steak (Ribeye)','Pound',12.99,'Premium cuts of
Angus beef ribeye, perfect for grilling.',40,15),(4,2,'Angus Beef
Steak (Sirloin)','Pound',10.99,'Premium cuts of Angus beef sirloin,
perfect for grilling.',35,10),(5,1,'Spaghetti Pasta (1kg)','Pack'
,1.49,'Classic Italian spaghetti pasta made from durum wheat semolina
.',150,40),(5,2,'Spaghetti Pasta (500g)','Pack',0.99,'Classic Italian
 spaghetti pasta made from durum wheat semolina, in a smaller pack.'
,180,45),(6,1,'Natural Yogurt (500ml)','Bottle',1.99,'Creamy yogurt
with live probiotics and no added sugar.',120,30),(6,2,'Natural
Yogurt (1L)','Bottle',3.49,'Creamy yogurt with live probiotics and no
 added sugar, in a larger bottle.',90,20),(7,1,'Almond Milk (200ml
boxed)','Box',1.29,'Dairy-free milk alternative made from real
almonds, in a convenient small box.',200,60),(7,2,'Almond Milk (500ml
 boxed)','Box',2.49,'Dairy-free milk alternative made from real
almonds, in a larger box.',150,40),(8,1,'Cage-Free Brown Eggs (Dozen)
','Dozen',2.99,'Nutritious eggs from hens raised in cage-free
environments, packed by the dozen.',100,25),(8,2,'Cage-Free Brown
Eggs (Half Dozen)','Pack',1.79,'Nutritious eggs from hens raised in
cage-free environments, packed by the half dozen.',120,35),(9,1,'
Organic Spinach (Bunch)','Bunch',2.99,'Fresh organic spinach, washed
and ready to eat, sold as a bunch.',80,20),(9,2,'Organic Spinach (
Prepackaged)','Bag',3.49,'Prepackaged fresh organic spinach, washed
and ready to eat.',70,30),(10,1,'Ripe Avocados (Single)','Each',1.49,
'Single rich and creamy avocado, great for guacamole.',180,40),(10,2,
'Ripe Avocados (Bag of 4)','Bag',5.49,'A bag of 4 rich and creamy
avocados, great for guacamole.',90,20),(11,1,'Cheddar Cheese (200g)',
'Block',3.99,'Rich and creamy aged cheddar cheese.',85,25),(11,2,'
Cheddar Cheese (500g)','Block',7.99,'Rich and creamy aged cheddar
cheese in a larger block.',65,15),(12,1,'Greek Yogurt (150g)','Cup'
,0.99,'Thick and creamy yogurt with a hint of tartness, in a single
serving cup.',110,50),(12,2,'Greek Yogurt (500g)','Tub',3.49,'Thick
and creamy yogurt with a hint of tartness, in a larger tub.',80,35)
,(13,1,'Organic Milk (1L)','Bottle',2.99,'Organic milk from grass-fed
 cows, in a 1L bottle.',100,40),(13,2,'Organic Milk (2L)','Bottle'
,4.99,'Organic milk from grass-fed cows, in a more economical 2L
bottle.',75,30),(14,1,'Espresso Coffee Beans (250g)','Bag',4.99,'Dark
 roasted espresso beans with a rich, bold flavor, in a 250g bag.'
,120,45),(14,2,'Espresso Coffee Beans (500g)','Bag',8.99,'Dark
roasted espresso beans with a rich, bold flavor, in a 500g bag.'
,100,40),(15,1,'Green Tea (20 bags)','Box',3.49,'Refreshing green tea
 rich in antioxidants, comes in a box of 20 bags.',130,60),(15,2,'
Green Tea (Loose Leaf 100g)','Pack',4.99,'Refreshing loose-leaf green
 tea rich in antioxidants.',100,40),(16,1,'Mineral Water (500ml)','
Bottle',0.99,'Pure spring water with added minerals for taste in a
500ml bottle.',200,50),(16,2,'Mineral Water (1.5L)','Bottle',1.49,'
Pure spring water with added minerals for taste in a 1.5L bottle.'
```

```
,150,40),(17,1,'Quinoa (250g)','Bag',2.99,'Nutritious whole-grain
quinoa, gluten-free and high in protein in a 250g bag.',120,30)
,(17,2,'Quinoa (1kg)','Bag',5.99,'Nutritious whole-grain quinoa,
gluten-free and high in protein in a 1kg bag.',100,25),(18,1,'Brown
Rice (500g)','Bag',1.99,'Whole-grain brown rice with a nutty flavor
in a 500g bag.',150,60),(18,2,'Brown Rice (2kg)','Bag',4.99,'Whole-
grain brown rice with a nutty flavor in a 2kg bag.',100,20),(19,1,'
Raspberry Jam (250g)','Jar',3.49,'Jam made with ripe raspberries and
pure cane sugar in a 250g jar.',80,40),(19,2,'Raspberry Jam (500g)','
Jar',5.99,'Jam made with ripe raspberries and pure cane sugar in a
500g jar.',60,20),(20,1,'Corn Flakes (500g)','Box',2.99,'Crunchy corn
 flakes, a classic breakfast cereal in a 500g box.',130,45),(20,2,'
Corn Flakes (1kg)','Box',4.99,'Crunchy corn flakes, a classic
breakfast cereal in a 1kg box.',70,30),(21,1,'Maple Syrup (250ml)','
Bottle',7.99,'Pure maple syrup, perfect for pancakes and waffles in a
 250ml bottle.',50,25),(21,2,'Maple Syrup (500ml)','Bottle',14.99,'
Pure maple syrup, perfect for pancakes and waffles in a 500ml bottle.
',30,15),(22,1,'Chicken Breasts (500g)','Package',4.99,'Boneless and
skinless chicken breasts, versatile for any dish in a 500g package.'
,200,100),(22,2,'Chicken Breasts (1kg)','Package',9.49,'Boneless and
skinless chicken breasts, versatile for any dish in a 1kg package.'
,150,75),(23,1,'Pork Chops (500g)','Package',5.99,'Juicy and tender
pork chops, ready for the grill in a 500g package.',150,50),(23,2,'
Pork Chops (1kg)','Package',11.49,'Juicy and tender pork chops, ready
 for the grill in a 1kg package.',100,40),(24,1,'Lamb Shoulder (1kg)'
,'Package',14.99,'Rich and flavorful lamb shoulder, ideal for slow
cooking in a 1kg package.',90,45),(24,2,'Lamb Shoulder (2kg)','
Package',28.99,'Rich and flavorful lamb shoulder, ideal for slow
cooking in a 2kg package.',50,20),(25,1,'Tilapia Fillets (500g)','
Package',6.99,'Mild-flavored tilapia, perfect for quick and healthy
meals in a 500g package.',120,60),(25,2,'Tilapia Fillets (1kg)','
Package',13.49,'Mild-flavored tilapia, perfect for quick and healthy
meals in a 1kg package.',90,45),(26,1,'Shrimp (250g)','Package',8.99,
'Fresh shrimp, cleaned and deveined, ready to cook in a 250g package.
',100,50),(26,2,'Shrimp (500g)','Package',17.49,'Fresh shrimp,
cleaned and deveined, ready to cook in a 500g package.',80,40),(27,1,
'Cod Fish (500g)','Package',7.99,'Flaky and mild white fish, perfect
for fish and chips in a 500g package.',110,55),(27,2,'Cod Fish (1kg)'
,'Package',15.49,'Flaky and mild white fish, perfect for fish and
chips in a 1kg package.',70,35),(28,1,'Kale (200g)','Bag',2.49,'
Nutrient-dense kale, great for salads and smoothies in a 200g bag.'
,150,75),(28,2,'Kale (500g)','Bag',4.99,'Nutrient-dense kale, great
for salads and smoothies in a 500g bag.',120,60),(29,1,'Sweet
Potatoes (1kg)','Bag',3.49,'Versatile sweet potatoes, rich in
vitamins and fiber in a 1kg bag.',130,65),(29,2,'Sweet Potatoes (2kg)
','Bag',6.49,'Versatile sweet potatoes, rich in vitamins and fiber in
 a 2kg bag.',100,50),(30,1,'Baby Carrots (500g)','Bag',1.99,'
Convenient baby carrots, peeled and ready to snack on in a 500g bag.'
,200,100),(30,2,'Baby Carrots (1kg)','Bag',3.49,'Convenient baby
```

```
carrots, peeled and ready to snack on in a 1kg bag.',150,75),(31,1,'
Fuji Apples Bag','Bag',3.99,'5lb bag of crisp Fuji apples',100,10)
,(31,2,'Fuji Apples Loose','Each',0.79,'Single Fuji apple',200,30)
,(32,1,'Bananas Bunch','Bunch',1.29,'Bunch of ripe bananas',120,15)
,(32,2,'Bananas Single','Each',0.19,'Single ripe banana',180,40)
,(33,1,'Blueberries Pack','Pack',2.99,'Pack of plump and sweet
blueberries',150,20),(33,2,'Blueberries Bulk','lb',5.99,'Bulk
blueberries for baking',60,5),(34,1,'Rye Bread Loaf','Loaf',3.49,'
Loaf of hearty rye bread',80,10),(34,2,'Rye Bread Sliced','Sliced
Loaf',3.99,'Sliced rye bread loaf',85,10),(35,1,'Croissants Pack of 6
','Pack',4.99,'Pack of 6 fresh-baked croissants',70,15),(35,2,'
Croissant Single','Each',0.99,'Single buttery croissant',150,40)
,(36,1,'Plain Bagels Pack','Pack',3.49,'Pack of 6 chewy plain bagels'
,90,15),(36,2,'Everything Bagel','Each',0.69,'Single \'everything\'
flavored bagel',110,30),(37,1,'Chocolate Chip Cookies Dozen','Dozen'
,3.99,'Dozen of classic chocolate chip cookies',60,15),(37,2,'
Chocolate Chip Cookie Single','Each',0.33,'Single chocolate chip
cookie',180,50),(38,1,'Pretzels Large Bag','Bag',2.49,'Large bag of
salty pretzels',100,25),(38,2,'Pretzels Snack Pack','Pack',0.99,'
Snack pack of crunchy pretzels',200,60),(39,1,'Almonds Roasted','Bag'
,6.99,'Roasted almonds bag',80,20),(39,2,'Almonds Raw','lb',7.99,'Raw
 almonds bulk',50,10),(40,1,'Olive Oil 500ml','Bottle',5.99,'500ml
bottle of extra virgin olive oil',100,20),(40,2,'Olive Oil 1L','
Bottle',10.99,'1L bottle of extra virgin olive oil',50,10),(41,1,'
Balsamic Vinegar 250ml','Bottle',4.99,'250ml bottle of aged balsamic
vinegar',80,20),(41,2,'Balsamic Vinegar 500ml','Bottle',8.99,'500ml
bottle of aged balsamic vinegar',40,10),(42,1,'Sea Salt Grinder','
Each',2.99,'Grinder with natural sea salt',120,30),(42,2,'Sea Salt
Bulk','lb',1.99,'Bulk natural sea salt',60,15),(43,1,'Pepperoni Pizza
 Large','Box',7.99,'Large ready-to-bake pepperoni pizza',50,10)
,(43,2,'Pepperoni Pizza Personal','Box',3.99,'Personal size pepperoni
 pizza',80,20),(44,1,'Ice Cream Vanilla','Pint',3.49,'Vanilla ice
cream pint',130,25),(44,2,'Ice Cream Chocolate','Pint',3.49,'
Chocolate ice cream pint',130,25),(45,1,'Frozen Peas 1lb','Bag',1.99,
'1lb bag of frozen green peas',100,20),(45,2,'Frozen Peas 2lb','Bag'
,3.49,'2lb bag of frozen green peas',60,15),(46,1,'Granola Bars Small
 Pack','Pack',2.99,'Small pack of 5 granola bars',120,25),(46,2,'
Granola Bars Large Box','Box',4.99,'Large box containing 12 granola
bars',80,15),(47,1,'Rice Cakes Original','Pack',1.99,'Original flavor
 rice cakes',150,35),(47,2,'Rice Cakes Apple Cinnamon','Pack',2.49,'
Apple cinnamon flavor rice cakes',100,20),(48,1,'Peanut Butter Smooth
','Jar',3.49,'Smooth peanut butter in a jar',110,30),(48,2,'Peanut
Butter Crunchy','Jar',3.49,'Crunchy peanut butter in a jar',120,30)
,(49,1,'Honey Bottle Small','Bottle',4.99,'Small bottle of natural
honey',90,20),(49,2,'Honey Bottle Large','Bottle',7.99,'Large bottle
of natural honey',60,15),(50,1,'Canned Tomatoes Diced','Can',1.29,'
Diced canned tomatoes',200,50),(50,2,'Canned Tomatoes Whole','Can'
,1.29,'Whole canned tomatoes',180,40),(51,1,'Canned Tuna in Water','
Can',1.49,'Chunk light tuna in water',160,40),(51,2,'Canned Tuna in
```

```
   Oil','Can',1.69,'Chunk light tuna in oil',150,35),(52,1,'Spicy Salsa
   Jar','Jar',2.99,'Jar of zesty salsa with a kick',130,30),(52,2,'Spicy
    Salsa Bulk','lb',5.99,'Bulk zesty salsa for catering',50,10),(53,1,'
   BBQ Sauce Bottle','Bottle',2.99,'Rich and smoky BBQ sauce',120,20)
   ,(53,2,'BBQ Sauce Squeeze','Bottle',3.49,'Squeeze bottle of BBQ sauce
   ',100,25),(54,1,'Mustard Classic','Bottle',1.99,'Classic yellow
   mustard bottle',140,40),(54,2,'Mustard Spicy Brown','Bottle',2.29,'
   Spicy brown mustard bottle',120,30),(55,1,'Ketchup Bottle','Bottle'
   ,2.49,'Tomato ketchup bottle',160,35),(55,2,'Ketchup Squeeze Pack','
   Pack',0.99,'Squeeze pack of tomato ketchup',200,60),(56,1,'Soy Sauce
   Bottle Small','Bottle',2.99,'Small bottle of traditional soy sauce'
   ,130,25),(56,2,'Soy Sauce Bottle Large','Bottle',5.49,'Large bottle
   of traditional soy sauce',70,15),(57,1,'Green Olives Jar','Jar',3.99,
   'Jar of pitted green olives',110,20),(57,2,'Green Olives Bulk','lb'
   ,6.99,'Bulk pitted green olives',50,10),(58,1,'Black Beans Can','Can'
   ,1.49,'Can of ready-to-use black beans',180,45),(58,2,'Black Beans
   Dry Bulk','lb',2.49,'Bulk dry black beans',60,15),(59,1,'Coconut
   Water Small','Bottle',1.99,'Small bottle of hydrating coconut water'
   ,120,30),(59,2,'Coconut Water Large','Bottle',3.49,'Large bottle of
   hydrating coconut water',80,20),(60,1,'Soy Milk Carton','Carton'
   ,2.99,'Carton of dairy-free soy milk',130,25),(60,2,'Soy Milk Bottle'
   ,'Bottle',4.99,'Bottle of dairy-free soy milk',90,20);
/*!40000 ALTER TABLE `t_product_variants` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_products`
--

DROP TABLE IF EXISTS `t_products`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_products` (
  `pk_product_id` int NOT NULL,
  `product_name` varchar(255) DEFAULT NULL,
  `category` varchar(255) DEFAULT NULL,
  `description` text,
  PRIMARY KEY (`pk_product_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_products`
--

LOCK TABLES `t_products` WRITE;
/*!40000 ALTER TABLE `t_products` DISABLE KEYS */;
INSERT INTO `t_products` VALUES (1,'Organic Red Apples','Fruits','Crisp
   and sweet organic apples from local orchards.'),(2,'Whole Wheat Bread
```

```
','Bakery','Freshly baked bread made with 100% whole wheat flour.')
,(3,'Atlantic Salmon','Seafood','Fresh Atlantic salmon, rich in Omega
-3 fatty acids.'),(4,'Angus Beef Steak','Meat','Premium cuts of Angus
 beef, perfect for grilling.'),(5,'Spaghetti Pasta','Pasta & Rice','
Classic Italian pasta made from durum wheat semolina.'),(6,'Natural
Yogurt','Dairy','Creamy yogurt with live probiotics and no added
sugar.'),(7,'Almond Milk','Beverages','Dairy-free milk alternative
made from real almonds.'),(8,'Cage-Free Brown Eggs','Eggs','
Nutritious eggs from hens raised in cage-free environments.'),(9,'
Organic Spinach','Vegetables','Fresh organic spinach, washed and
ready to eat.'),(10,'Ripe Avocados','Produce','Rich and creamy
avocados, great for guacamole.'),(11,'Cheddar Cheese','Dairy','Rich
and creamy aged cheddar cheese.'),(12,'Greek Yogurt','Dairy','Thick
and creamy yogurt with a hint of tartness.'),(13,'Organic Milk','
Dairy','Organic milk from grass-fed cows.'),(14,'Espresso Coffee
Beans','Beverages','Dark roasted beans with a rich, bold flavor.')
,(15,'Green Tea','Beverages','Refreshing green tea rich in
antioxidants.'),(16,'Mineral Water','Beverages','Pure spring water
with added minerals for taste.'),(17,'Quinoa','Pasta & Rice','
Nutritious whole-grain quinoa, gluten-free and high in protein.')
,(18,'Brown Rice','Pasta & Rice','Whole-grain brown rice with a nutty
 flavor.'),(19,'Raspberry Jam','Breakfast Foods','Jam made with ripe
raspberries and pure cane sugar.'),(20,'Corn Flakes','Breakfast Foods
','Crunchy corn flakes, a classic breakfast cereal.'),(21,'Maple
Syrup','Breakfast Foods','Pure maple syrup, perfect for pancakes and
waffles.'),(22,'Chicken Breasts','Meat','Boneless and skinless
chicken breasts, versatile for any dish.'),(23,'Pork Chops','Meat','
Juicy and tender pork chops, ready for the grill.'),(24,'Lamb
Shoulder','Meat','Rich and flavorful lamb shoulder, ideal for slow
cooking.'),(25,'Tilapia Fillets','Seafood','Mild-flavored tilapia,
perfect for quick and healthy meals.'),(26,'Shrimp','Seafood','Fresh
shrimp, cleaned and deveined, ready to cook.'),(27,'Cod Fish','
Seafood','Flaky and mild white fish, perfect for fish and chips.')
,(28,'Kale','Vegetables','Nutrient-dense kale, great for salads and
smoothies.'),(29,'Sweet Potatoes','Vegetables','Versatile sweet
potatoes, rich in vitamins and fiber.'),(30,'Baby Carrots','
Vegetables','Convenient baby carrots, peeled and ready to snack on.')
,(31,'Fuji Apples','Fruits','Crisp and juicy Fuji apples with a
balanced sweet-tart flavor.'),(32,'Bananas','Fruits','Ripe bananas,
full of potassium and perfect for on-the-go.'),(33,'Blueberries','
Fruits','Plump and sweet blueberries, ideal for baking or snacking.')
,(34,'Rye Bread','Bakery','Hearty rye bread with a distinctive flavor
, perfect for sandwiches.'),(35,'Croissants','Bakery','Flaky and
buttery croissants, baked fresh daily.'),(36,'Bagels','Bakery','Chewy
 bagels, available in various flavors.'),(37,'Chocolate Chip Cookies'
,'Snacks','Classic cookies with rich chocolate chips.'),(38,'Pretzels
','Snacks','Salty and crunchy pretzels, a perfect snack anytime.')
,(39,'Almonds','Snacks','Whole almonds, a healthy and satisfying
snack.'),(40,'Olive Oil','Cooking Essentials','Extra virgin olive oil
```

```
    with a fruity flavor profile.'),(41,'Balsamic Vinegar','Cooking
   Essentials','Aged balsamic vinegar, ideal for dressings and marinades
   .'),(42,'Sea Salt','Cooking Essentials','Natural sea salt, perfect
   for seasoning any dish.'),(43,'Pepperoni Pizza','Frozen Foods','Ready
   -to-bake pepperoni pizza with a crispy crust.'),(44,'Ice Cream','
   Frozen Foods','Creamy ice cream in various flavors for a sweet treat.
   '),(45,'Frozen Peas','Frozen Foods','Frozen green peas, a convenient
   and healthy side dish.'),(46,'Granola Bars','Snacks','Hearty granola
   bars made with whole grains and honey.'),(47,'Rice Cakes','Snacks','
   Light and airy rice cakes, a guilt-free snacking option.'),(48,'
   Peanut Butter','Pantry Items','Smooth peanut butter, made with
   roasted peanuts.'),(49,'Honey','Pantry Items','Natural honey, perfect
    as a sweetener or in recipes.'),(50,'Canned Tomatoes','Pantry Items'
   ,'Diced canned tomatoes, a versatile pantry staple.'),(51,'Canned
   Tuna','Pantry Items','Chunk light tuna in water, great for sandwiches
    and salads.'),(52,'Spicy Salsa','Pantry Items','Zesty salsa with a
   kick, perfect for dipping or as a condiment.'),(53,'BBQ Sauce','
   Pantry Items','Rich and smoky BBQ sauce, ideal for grilling.'),(54,'
   Mustard','Pantry Items','Classic yellow mustard, a must-have for hot
   dogs and burgers.'),(55,'Ketchup','Pantry Items','Tomato ketchup with
    a perfect balance of sweet and tangy.'),(56,'Soy Sauce','Pantry
   Items','Traditional soy sauce, a staple for Asian cuisine.'),(57,'
   Green Olives','Pantry Items','Pitted green olives, great as a snack
   or in recipes.'),(58,'Black Beans','Pantry Items','Canned black beans
   , ready to use in soups or salads.'),(59,'Coconut Water','Beverages',
   'Hydrating coconut water, rich in electrolytes.'),(60,'Soy Milk','
   Beverages','Dairy-free soy milk, a great source of protein.');
/*!40000 ALTER TABLE `t_products` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_reduction_promotions`
--

DROP TABLE IF EXISTS `t_reduction_promotions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_reduction_promotions` (
  `pk_promotion_id` int NOT NULL,
  `threshold_amount` float DEFAULT NULL,
  `discount_amount` float DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`pk_promotion_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_reduction_promotions`
```

```
--

LOCK TABLES `t_reduction_promotions` WRITE;
/*!40000 ALTER TABLE `t_reduction_promotions` DISABLE KEYS */;
INSERT INTO `t_reduction_promotions` VALUES (1,100,10,'2023-04-01','
    2023-04-10'),(2,200,20,'2023-04-11','2023-04-20'),(3,300,30,'
    2023-04-21','2023-04-30'),(4,150,15,'2023-05-01','2023-05-10')
    ,(5,250,25,'2023-05-11','2023-05-20'),(6,350,35,'2023-05-21','
    2023-05-30'),(7,120,12,'2023-06-01','2023-06-10'),(8,220,22,'
    2023-06-11','2023-06-20'),(9,320,32,'2023-06-21','2023-06-30')
    ,(10,180,18,'2023-07-01','2023-07-10');
/*!40000 ALTER TABLE `t_reduction_promotions` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_sales_reports`
--

DROP TABLE IF EXISTS `t_sales_reports`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_sales_reports` (
  `pk_sales_report_id` int NOT NULL,
  `report_type` varchar(255) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `total_sales` float DEFAULT NULL,
  `total_revenue` float DEFAULT NULL,
  `create_time` datetime DEFAULT NULL,
  PRIMARY KEY (`pk_sales_report_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_sales_reports`
--

LOCK TABLES `t_sales_reports` WRITE;
/*!40000 ALTER TABLE `t_sales_reports` DISABLE KEYS */;
INSERT INTO `t_sales_reports` VALUES (1,'Weekly','2023-03-01','
    2023-03-07',12345.5,15342.8,'2023-03-08 09:00:00'),(2,'Monthly','
    2023-02-01','2023-02-28',23456.8,28906.5,'2023-03-01 09:00:00'),(3,'
    Annual','2022-01-01','2022-12-31',34567.9,42500,'2023-01-01 09:00:00'
    ),(4,'Quarterly','2023-01-01','2023-03-31',45678.1,56007.8,'
    2023-04-01 09:00:00'),(5,'Weekly','2023-03-08','2023-03-14'
    ,56789.2,69854.2,'2023-03-15 09:00:00'),(6,'Monthly','2023-03-01','
    2023-03-31',12345.5,15000,'2023-04-01 09:00:00'),(7,'Weekly','
    2023-03-15','2023-03-21',23456.8,28765.8,'2023-03-22 09:00:00'),(8,'
    Quarterly','2022-10-01','2022-12-31',34567.9,42345.6,'2023-01-01
```

```
    09:00:00'),(9,'Annual','2021-01-01','2021-12-31',45678.1,55987.9,'
    2022-01-01 09:00:00'),(10,'Weekly','2023-03-22','2023-03-28'
    ,56789.2,69012.5,'2023-03-29 09:00:00');
/*!40000 ALTER TABLE `t_sales_reports` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_supplier_contacts`
--

DROP TABLE IF EXISTS `t_supplier_contacts`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_supplier_contacts` (
  `pk_contact_id` int NOT NULL,
  `fk_supplier_id` int DEFAULT NULL,
  `contact_name` varchar(255) DEFAULT NULL,
  `contact_title` varchar(255) DEFAULT NULL,
  `phone_number` int DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`pk_contact_id`),
  KEY `fk_supplier_id` (`fk_supplier_id`),
  CONSTRAINT `t_supplier_contacts_ibfk_1` FOREIGN KEY (`fk_supplier_id`)
    REFERENCES `t_suppliers` (`pk_supplier_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_supplier_contacts`
--

LOCK TABLES `t_supplier_contacts` WRITE;
/*!40000 ALTER TABLE `t_supplier_contacts` DISABLE KEYS */;
INSERT INTO `t_supplier_contacts` VALUES (1,101,'John Doe','Sales
    Representative',1234567890,'johndoe@email.com'),(2,102,'Jane Smith','
    Account Manager',1234567891,'janesmith@email.com'),(3,103,'Michael
    Brown','Customer Service',1234567892,'michaelb@email.com'),(4,104,'
    Lisa White','Quality Assurance',1234567893,'lisaw@email.com'),(5,105,
    'Mark Jones','Procurement Manager',1234567894,'markj@email.com')
    ,(6,106,'Emily Davis','Logistics Coordinator',1234567895,'
    emilyd@email.com'),(7,107,'David Wilson','Marketing Specialist'
    ,1234567896,'davidw@email.com'),(8,108,'Sarah Miller','Production
    Supervisor',1234567897,'sarahm@email.com'),(9,109,'James Taylor','
    Technical Support',1234567898,'jamest@email.com'),(10,110,'Laura
    Anderson','Operations Manager',1234567899,'lauraa@email.com');
/*!40000 ALTER TABLE `t_supplier_contacts` ENABLE KEYS */;
UNLOCK TABLES;

--
```

```
-- Table structure for table `t_suppliers`
--

DROP TABLE IF EXISTS `t_suppliers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_suppliers` (
  `pk_supplier_id` int NOT NULL,
  `company_name` varchar(255) DEFAULT NULL,
  `company_address` varchar(255) DEFAULT NULL,
  `company_website` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`pk_supplier_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_suppliers`
--

LOCK TABLES `t_suppliers` WRITE;
/*!40000 ALTER TABLE `t_suppliers` DISABLE KEYS */;
INSERT INTO `t_suppliers` VALUES (101,'Fresh Farm Produce','123 Country
    Road, Ruralville','www.freshfarmproduce.com'),(102,'Oceanic Seafoods'
    ,'456 Coastal Avenue, Seaport','www.oceanicseafoods.com'),(103,'Grand
     Grains Ltd','789 Wheatfield Blvd, Farmtown','www.grandgrains.com')
    ,(104,'Dairy Delights Inc','321 Milky Way, Dairyland','www.
    dairydelightsinc.com'),(105,'Poultry Providers','654 Cluck Street,
    Chickenville','www.poultryproviders.com'),(106,'Quality Quenchers','
    987 Thirsty Thoroughfare, Beverageburg','www.qualityquenchers.com')
    ,(107,'Crispy Crunch Snacks','269 Snack Avenue, Snacktown','www.
    crispycrunchsnacks.com'),(108,'Bakery Bliss','852 Freshloaf Road,
    Bakersville','www.bakerybliss.com'),(109,'Select Meats','741 Steak
    Street, Meatsville','www.selectmeats.com'),(110,'Farm Fresh Eggs','
    963 Yolk Road, Eggstown','www.farmfresheggs.com');
/*!40000 ALTER TABLE `t_suppliers` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_supply_records`
--

DROP TABLE IF EXISTS `t_supply_records`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_supply_records` (
  `pk_supply_record_id` int NOT NULL,
  `fk_supply_product_id` int DEFAULT NULL,
  `fk_supply_variant_id` int DEFAULT NULL,
  `fk_supplier_id` int DEFAULT NULL,
```

```
348   `supply_date` date DEFAULT NULL,
349   `supply_quantity` float DEFAULT NULL,
350   `total_price` float DEFAULT NULL,
351   `pay_term` text,
352   PRIMARY KEY (`pk_supply_record_id`),
353   KEY `fk_supply_product_id` (`fk_supply_product_id`,`
       fk_supply_variant_id`),
354   KEY `fk_supplier_id` (`fk_supplier_id`),
355   CONSTRAINT `t_supply_records_ibfk_1` FOREIGN KEY (`
       fk_supply_product_id`, `fk_supply_variant_id`) REFERENCES `
       t_product_variants` (`pk_product_id`, `pk_variant_id`),
356   CONSTRAINT `t_supply_records_ibfk_2` FOREIGN KEY (`fk_supplier_id`)
       REFERENCES `t_suppliers` (`pk_supplier_id`)
357 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
358 /*!40101 SET character_set_client = @saved_cs_client */;
359
360 --
361 -- Dumping data for table `t_supply_records`
362 --
363
364 LOCK TABLES `t_supply_records` WRITE;
365 /*!40000 ALTER TABLE `t_supply_records` DISABLE KEYS */;
366 INSERT INTO `t_supply_records` VALUES (1,1,1,101,'2023-04-01',150,890,'
    Net 30 days'),(2,1,2,102,'2023-04-02',200,130,'Net 30 days')
    ,(3,2,1,103,'2023-04-03',100,250,'Net 45 days'),(4,2,2,104,'
    2023-04-04',120,400,'Net 45 days'),(5,3,1,105,'2023-04-05',60,500,'
    Net 30 days'),(6,3,2,106,'2023-04-06',40,680,'Net 60 days')
    ,(7,4,1,107,'2023-04-07',50,600,'Net 30 days'),(8,4,2,108,'2023-04-08
    ',45,450,'Net 30 days'),(9,5,1,109,'2023-04-09',160,230,'Net 45 days'
    ),(10,5,2,110,'2023-04-10',180,170,'Net 45 days'),(11,6,1,101,'
    2023-04-11',130,250,'Net 60 days'),(12,6,2,102,'2023-04-12',100,340,'
    Net 60 days'),(13,7,1,103,'2023-04-13',210,300,'Net 30 days')
    ,(14,7,2,104,'2023-04-14',160,390,'Net 30 days'),(15,8,1,105,'
    2023-04-15',110,320,'Net 45 days'),(16,8,2,106,'2023-04-16',130,240,'
    Net 45 days'),(17,9,1,107,'2023-04-17',90,260,'Net 60 days')
    ,(18,9,2,108,'2023-04-18',80,280,'Net 60 days'),(19,10,1,109,'
    2023-04-19',190,350,'Net 30 days'),(20,10,2,110,'2023-04-20',100,490,
    'Net 30 days'),(21,11,1,101,'2023-04-21',90,850,'Net 45 days')
    ,(22,11,2,102,'2023-04-22',70,660,'Net 45 days'),(23,12,1,103,'
    2023-04-23',120,110,'Net 60 days'),(24,12,2,104,'2023-04-24',85,300,'
    Net 60 days'),(25,13,1,105,'2023-04-25',105,500,'Net 30 days')
    ,(26,13,2,106,'2023-04-26',80,390,'Net 30 days'),(27,14,1,107,'
    2023-04-27',130,1200,'Net 45 days'),(28,14,2,108,'2023-04-28'
    ,110,990,'Net 45 days'),(29,15,1,109,'2023-04-29',140,460,'Net 60
    days'),(30,15,2,110,'2023-04-30',120,480,'Net 60 days'),(31,1,1,101,'
    2023-05-01',150,870,'Net 30 days'),(32,2,1,102,'2023-05-02',100,290,'
    Net 45 days'),(33,3,2,103,'2023-05-03',45,720,'Net 60 days')
    ,(34,4,1,104,'2023-05-04',55,650,'Net 30 days'),(35,5,2,105,'
    2023-05-05',185,175,'Net 45 days'),(36,6,1,106,'2023-05-06',135,260,'
```

```
Net 60 days'),(37,7,2,107,'2023-05-07',155,385,'Net 30 days')
,(38,8,1,108,'2023-05-08',115,315,'Net 45 days'),(39,9,2,109,'
2023-05-09',85,275,'Net 60 days'),(40,10,1,110,'2023-05-10',195,360,'
Net 30 days'),(41,16,1,101,'2023-04-01',250,174.3,'Net 30')
,(42,16,2,102,'2023-04-02',200,208.6,'Net 30'),(43,17,1,103,'
2023-04-03',150,314.1,'Net 60'),(44,17,2,104,'2023-04-04',120,503.64,
'Net 60'),(45,18,1,105,'2023-04-05',180,251.58,'Net 30')
,(46,18,2,106,'2023-04-06',100,349.3,'Net 30'),(47,19,1,107,'
2023-04-07',90,222.57,'Net 30'),(48,19,2,108,'2023-04-08',80,335.16,'
Net 60'),(49,20,1,109,'2023-04-09',130,271.53,'Net 30'),(50,20,2,110,
'2023-04-10',70,244.93,'Net 60'),(51,21,1,101,'2023-04-11',60,335.58,
'Net 30'),(52,21,2,102,'2023-04-12',50,524.25,'Net 30'),(53,22,1,103,
'2023-04-13',220,307.86,'Net 60'),(54,22,2,104,'2023-04-14'
,160,529.18,'Net 60'),(55,23,1,105,'2023-04-15',170,419.58,'Net 30')
,(56,23,2,106,'2023-04-16',100,602.37,'Net 30'),(57,24,1,107,'
2023-04-17',95,629.65,'Net 30'),(58,24,2,108,'2023-04-18',55,1015.65,
'Net 60'),(59,25,1,109,'2023-04-19',130,461.13,'Net 30')
,(60,25,2,110,'2023-04-20',95,716.83,'Net 60'),(61,26,1,101,'
2023-04-21',110,559.89,'Net 30'),(62,26,2,102,'2023-04-22'
,85,1022.45,'Net 30'),(63,27,1,103,'2023-04-23',120,503.64,'Net 60')
,(64,27,2,104,'2023-04-24',75,808.58,'Net 60'),(65,28,1,105,'
2023-04-25',160,279.44,'Net 30'),(66,28,2,106,'2023-04-26'
,130,454.65,'Net 30'),(67,29,1,107,'2023-04-27',140,342.44,'Net 30')
,(68,29,2,108,'2023-04-28',105,454.65,'Net 60'),(69,30,1,109,'
2023-04-29',210,293.37,'Net 30'),(70,30,2,110,'2023-04-30'
,160,384.86,'Net 60'),(71,16,1,101,'2023-05-01',260,182.28,'Net 30')
,(72,17,2,102,'2023-05-02',130,545.79,'Net 30'),(73,18,1,103,'
2023-05-03',190,269.73,'Net 60'),(74,19,2,104,'2023-05-04',85,359.19,
'Net 60'),(75,20,1,105,'2023-05-05',135,284.85,'Net 30')
,(76,31,1,101,'2023-04-01',150,419.85,'Net 30'),(77,31,2,102,'
2023-04-02',200,110.6,'Net 30'),(78,32,1,103,'2023-04-03',150,135.45,
'Net 60'),(79,32,2,104,'2023-04-04',180,23.94,'Net 30'),(80,33,1,105,
'2023-04-05',200,419.3,'Net 60'),(81,33,2,106,'2023-04-06',60,251.58,
'Net 30'),(82,34,1,107,'2023-04-07',100,244.3,'Net 60'),(83,34,2,108,
'2023-04-08',85,236.25,'Net 30'),(84,35,1,109,'2023-04-09',80,279.3,'
Net 30'),(85,35,2,110,'2023-04-10',150,104.65,'Net 60'),(86,36,1,101,
'2023-04-11',100,244.3,'Net 30'),(87,36,2,102,'2023-04-12',110,53.13,
'Net 60'),(88,37,1,103,'2023-04-13',70,195.3,'Net 30'),(89,37,2,104,'
2023-04-14',180,42.84,'Net 60'),(90,38,1,105,'2023-04-15',120,174.3,'
Net 30'),(91,38,2,106,'2023-04-16',200,138.6,'Net 60'),(92,39,1,107,'
2023-04-17',90,440.1,'Net 30'),(93,39,2,108,'2023-04-18',50,279.3,'
Net 60'),(94,40,1,109,'2023-04-19',120,419.58,'Net 30'),(95,40,2,110,
'2023-04-20',60,461.7,'Net 60'),(96,41,1,101,'2023-04-21',100,349.3,'
Net 30'),(97,41,2,102,'2023-04-22',50,314.65,'Net 60'),(98,42,1,103,'
2023-04-23',140,293.58,'Net 30'),(99,42,2,104,'2023-04-24',70,97.3,'
Net 60'),(100,43,1,105,'2023-04-25',60,335.58,'Net 30')
,(101,43,2,106,'2023-04-26',90,251.1,'Net 60'),(102,44,1,107,'
2023-04-27',150,363.75,'Net 30'),(103,44,2,108,'2023-04-28'
,140,343.58,'Net 60'),(104,45,1,109,'2023-04-29',120,139.86,'Net 30')
```

```
    ,(105,45,2,110,'2023-04-30',70,172.55,'Net 60'),(106,31,1,101,'
    2023-05-01',160,447.84,'Net 30'),(107,32,2,102,'2023-05-02'
    ,190,112.84,'Net 60'),(108,33,1,103,'2023-05-03',170,398.65,'Net 30')
    ,(109,34,2,104,'2023-05-04',95,265.65,'Net 60'),(110,35,1,105,'
    2023-05-05',75,262.5,'Net 30'),(111,46,1,101,'2023-03-10',100,209.3,'
    Net 30 days'),(112,46,2,102,'2023-03-11',80,279.3,'Net 30 days')
    ,(113,47,1,103,'2023-03-12',150,209.65,'Net 45 days'),(114,47,2,104,'
    2023-03-13',100,174.3,'Net 45 days'),(115,48,1,105,'2023-03-14'
    ,110,240.45,'Net 60 days'),(116,48,2,106,'2023-03-15',120,240.45,'Net
     60 days'),(117,49,1,107,'2023-03-16',90,314.1,'Net 30 days')
    ,(118,49,2,108,'2023-03-17',60,335.58,'Net 30 days'),(119,50,1,109,'
    2023-03-18',200,180.6,'Net 45 days'),(120,50,2,110,'2023-03-19'
    ,180,180.6,'Net 45 days'),(121,51,1,101,'2023-03-20',160,167.44,'Net
    60 days'),(122,51,2,102,'2023-03-21',150,176.85,'Net 60 days')
    ,(123,52,1,103,'2023-03-22',130,188.37,'Net 30 days'),(124,52,2,104,'
    2023-03-23',50,209.65,'Net 30 days'),(125,53,1,105,'2023-03-24'
    ,120,167.44,'Net 45 days'),(126,53,2,106,'2023-03-25',100,174.3,'Net
    45 days'),(127,54,1,107,'2023-03-26',140,195.72,'Net 60 days')
    ,(128,54,2,108,'2023-03-27',120,192.78,'Net 60 days'),(129,55,1,109,'
    2023-03-28',160,206.22,'Net 30 days'),(130,55,2,110,'2023-03-29'
    ,200,138.6,'Net 30 days'),(131,56,1,101,'2023-03-30',130,188.37,'Net
    45 days'),(132,56,2,102,'2023-03-31',70,229.95,'Net 45 days')
    ,(133,57,1,103,'2023-04-01',110,209.3,'Net 60 days'),(134,57,2,104,'
    2023-04-02',50,244.65,'Net 60 days'),(135,58,1,105,'2023-04-03'
    ,180,187.02,'Net 30 days'),(136,58,2,106,'2023-04-04',60,104.65,'Net
    30 days'),(137,59,1,107,'2023-04-05',120,139.86,'Net 45 days')
    ,(138,59,2,108,'2023-04-06',80,174.3,'Net 45 days'),(139,60,1,109,'
    2023-04-07',130,188.37,'Net 60 days'),(140,60,2,110,'2023-04-08'
    ,90,209.3,'Net 60 days');
/*!40000 ALTER TABLE `t_supply_records` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_transaction_details`
--

DROP TABLE IF EXISTS `t_transaction_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_transaction_details` (
  `pk_transaction_id` int NOT NULL,
  `pk_transaction_detail_id` int NOT NULL,
  `fk_product_id` int DEFAULT NULL,
  `fk_variant_id` int DEFAULT NULL,
  `purchasing_quantity` float DEFAULT NULL,
  `discounted_total_price` float DEFAULT NULL,
  PRIMARY KEY (`pk_transaction_id`,`pk_transaction_detail_id`),
  KEY `fk_product_id` (`fk_product_id`,`fk_variant_id`),
  CONSTRAINT `transaction_details_ibfk_1` FOREIGN KEY (`
```

```
    pk_transaction_id`) REFERENCES `t_transaction_records` (`
    pk_transaction_id`),
  CONSTRAINT `transaction_details_ibfk_2` FOREIGN KEY (`fk_product_id`,
    `fk_variant_id`) REFERENCES `t_product_variants` (`pk_product_id`, `
    pk_variant_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_transaction_details`
--

LOCK TABLES `t_transaction_details` WRITE;
/*!40000 ALTER TABLE `t_transaction_details` DISABLE KEYS */;
INSERT INTO `t_transaction_details` VALUES (1,1,1,1,2,11.98)
    ,(1,2,3,1,1,8.99),(2,1,4,2,3,32.97),(3,1,5,2,5,4.95),(3,2,6,1,2,3.98)
    ,(4,1,2,1,1,2.99),(4,2,7,2,2,4.98),(5,1,8,1,1,2.99),(5,2,10,2,1,5.49)
    ,(6,1,12,1,10,9.9),(6,2,9,2,3,10.47),(7,1,11,1,2,7.98)
    ,(8,1,13,2,1,4.99),(8,2,15,1,2,6.98),(9,1,2,2,4,13.96)
    ,(10,1,16,1,3,14.97),(10,2,3,2,1,3.33),(11,1,7,1,6,17.94)
    ,(12,1,1,2,2,5.98),(12,2,4,1,1,1.99),(13,1,8,2,10,29.9)
    ,(13,2,10,1,3,8.97),(14,1,12,2,4,3.96),(14,2,9,1,1,3.99)
    ,(15,1,11,2,5,9.95),(15,2,13,1,1,4.99),(16,1,2,2,3,14.97)
    ,(16,2,15,2,1,1.99),(17,1,16,2,2,9.98),(17,2,3,1,2,17.98);
/*!40000 ALTER TABLE `t_transaction_details` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `t_transaction_records`
--

DROP TABLE IF EXISTS `t_transaction_records`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `t_transaction_records` (
  `pk_transaction_id` int NOT NULL,
  `fk_responsible_employee_id` int DEFAULT NULL,
  `fk_customer_id` int DEFAULT NULL,
  `transaction_date` date DEFAULT NULL,
  `transaction_time` time DEFAULT NULL,
  `transaction_way` varchar(255) DEFAULT NULL,
  `fk_promotion_id` int DEFAULT NULL,
  `initial_amount` float DEFAULT NULL,
  `discounted_amount` float DEFAULT NULL,
  PRIMARY KEY (`pk_transaction_id`),
  KEY `fk_promotion_id` (`fk_promotion_id`),
  KEY `fk_customer_id` (`fk_customer_id`),
  KEY `fk_responsible_employee_id` (`fk_responsible_employee_id`),
  CONSTRAINT `transaction_records_ibfk_1` FOREIGN KEY (`fk_promotion_id
```

```
    ‘) REFERENCES ‘t_reduction_promotions‘ (‘pk_promotion_id‘),
  CONSTRAINT ‘transaction_records_ibfk_2‘ FOREIGN KEY (‘fk_customer_id‘)
     REFERENCES ‘t_customers‘ (‘pk_customer_id‘),
  CONSTRAINT ‘transaction_records_ibfk_3‘ FOREIGN KEY (‘
    fk_responsible_employee_id‘) REFERENCES ‘t_employees‘ (‘
    pk_employee_id‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table ‘t_transaction_records‘
--

LOCK TABLES ‘t_transaction_records‘ WRITE;
/*!40000 ALTER TABLE ‘t_transaction_records‘ DISABLE KEYS */;
INSERT INTO ‘t_transaction_records‘ VALUES (1,4,34,’2021-05-15’,’
    13:45:00’,’Online’,NULL,20.97,20.97),(2,17,76,’2021-05-16’,’10:30:00’
    ,’In-store’,NULL,32.97,32.97),(3,8,13,’2021-05-17’,’15:05:00’,’Online
    ’,NULL,8.93,8.93),(4,16,55,’2021-05-18’,’09:20:00’,’In-store’,NULL
    ,7.97,7.97),(5,3,83,’2021-05-19’,’14:55:00’,’Online’,NULL,8.48,8.48)
    ,(6,9,22,’2021-05-20’,’08:30:00’,’In-store’,NULL,20.37,20.37)
    ,(7,14,48,’2021-05-21’,’11:45:00’,’Online’,NULL,7.98,7.98),(8,5,69,’
    2021-05-22’,’16:15:00’,’In-store’,NULL,11.97,11.97),(9,12,38,’
    2021-05-23’,’10:00:00’,’Online’,NULL,13.96,13.96),(10,1,92,’
    2021-05-23’,’17:45:00’,’In-store’,NULL,18.3,18.3),(11,7,75,’
    2021-05-24’,’12:30:00’,’Online’,NULL,17.94,17.94),(12,2,57,’
    2021-05-25’,’16:00:00’,’In-store’,NULL,7.97,7.97),(13,19,65,’
    2021-05-26’,’09:20:00’,’Online’,NULL,38.87,38.87),(14,13,27,’
    2021-05-27’,’14:55:00’,’In-store’,NULL,7.95,7.95),(15,20,44,’
    2021-05-28’,’08:30:00’,’Online’,NULL,14.94,14.94),(16,11,32,’
    2021-05-29’,’11:45:00’,’In-store’,NULL,16.96,16.96),(17,6,89,’
    2021-05-30’,’16:15:00’,’Online’,NULL,27.96,27.96);
/*!40000 ALTER TABLE ‘t_transaction_records‘ ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table ‘t_variant_discounts‘
--

DROP TABLE IF EXISTS ‘t_variant_discounts‘;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE ‘t_variant_discounts‘ (
  ‘pk_variant_discount_id‘ int NOT NULL,
  ‘fk_discount_product_id‘ int DEFAULT NULL,
  ‘fk_discount_variant_id‘ int DEFAULT NULL,
  ‘variant_discount_rate‘ float DEFAULT NULL,
  ‘start_date‘ date DEFAULT NULL,
  ‘end_date‘ date DEFAULT NULL,
```

```
  PRIMARY KEY (`pk_variant_discount_id`),
  KEY `fk_discount_product_id` (`fk_discount_product_id`,`
    fk_discount_variant_id`),
  CONSTRAINT `t_variant_discounts_ibfk_1` FOREIGN KEY (`
    fk_discount_product_id`, `fk_discount_variant_id`) REFERENCES `
    t_product_variants` (`pk_product_id`, `pk_variant_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `t_variant_discounts`
--

LOCK TABLES `t_variant_discounts` WRITE;
/*!40000 ALTER TABLE `t_variant_discounts` DISABLE KEYS */;
INSERT INTO `t_variant_discounts` VALUES (1,1,1,0.1,'2023-04-01','
    2023-04-07'),(2,1,2,0.15,'2023-05-01','2023-05-15'),(3,2,1,0.05,'
    2023-04-10','2023-04-20'),(4,3,1,0.2,'2023-06-01','2023-06-15')
    ,(5,4,2,0.25,'2023-07-01','2023-07-10'),(6,5,1,0.1,'2023-08-01','
    2023-08-31'),(7,6,2,0.15,'2023-04-15','2023-05-01'),(8,7,1,0.05,'
    2023-05-20','2023-06-20'),(9,8,1,0.1,'2023-09-01','2023-09-15')
    ,(10,9,2,0.2,'2023-10-01','2023-10-07'),(11,10,1,0.12,'2023-11-01','
    2023-11-10'),(12,10,2,0.25,'2023-11-20','2023-12-05'),(13,11,1,0.08,'
    2023-09-15','2023-10-01'),(14,12,2,0.3,'2023-08-10','2023-08-25')
    ,(15,13,1,0.05,'2023-07-01','2023-07-20'),(16,13,2,0.1,'2023-04-20','
    2023-05-05'),(17,14,1,0.15,'2023-03-01','2023-03-31'),(18,15,1,0.2,'
    2023-06-15','2023-07-01'),(19,15,2,0.1,'2023-08-05','2023-08-20')
    ,(20,4,1,0.18,'2023-12-01','2023-12-15'),(21,16,1,0.1,'2023-04-01','
    2023-04-07'),(22,17,2,0.15,'2023-04-08','2023-04-15'),(23,18,1,0.05,'
    2023-04-16','2023-04-22'),(24,19,1,0.2,'2023-04-01','2023-04-07')
    ,(25,20,2,0.25,'2023-04-15','2023-04-20'),(26,21,1,0.3,'2023-04-10','
    2023-04-17'),(27,22,2,0.1,'2023-04-18','2023-04-25'),(28,23,1,0.15,'
    2023-04-05','2023-04-12'),(29,24,2,0.05,'2023-04-13','2023-04-19')
    ,(30,25,1,0.2,'2023-04-20','2023-04-30'),(31,26,2,0.1,'2023-04-01','
    2023-04-08'),(32,27,1,0.25,'2023-04-09','2023-04-15'),(33,28,2,0.15,'
    2023-04-16','2023-04-22'),(34,29,1,0.05,'2023-04-23','2023-04-29')
    ,(35,30,2,0.2,'2023-04-10','2023-04-17'),(36,16,2,0.1,'2023-04-18','
    2023-04-24'),(37,17,1,0.2,'2023-04-25','2023-04-30'),(38,18,2,0.15,'
    2023-04-01','2023-04-07'),(39,19,2,0.05,'2023-04-08','2023-04-14')
    ,(40,20,1,0.3,'2023-04-15','2023-04-21'),(41,31,1,0.1,'2023-04-01','
    2023-04-07'),(42,32,2,0.15,'2023-05-01','2023-05-08'),(43,33,1,0.05,'
    2023-06-01','2023-06-15'),(44,34,1,0.2,'2023-07-01','2023-07-07')
    ,(45,35,2,0.1,'2023-08-01','2023-08-10'),(46,36,1,0.25,'2023-09-01','
    2023-09-05'),(47,37,2,0.1,'2023-10-01','2023-10-06'),(48,38,1,0.2,'
    2023-11-01','2023-11-07'),(49,39,1,0.15,'2023-04-15','2023-04-22')
    ,(50,40,2,0.05,'2023-05-15','2023-05-20'),(51,41,1,0.2,'2023-06-16','
    2023-06-25'),(52,42,2,0.1,'2023-07-08','2023-07-14'),(53,43,1,0.15,'
    2023-08-11','2023-08-17'),(54,44,1,0.1,'2023-09-06','2023-09-12')
    ,(55,31,2,0.2,'2023-04-08','2023-04-14'),(56,32,1,0.3,'2023-05-09','
```

```
    2023-05-16'),(57,33,2,0.1,'2023-06-26','2023-06-30'),(58,34,2,0.15,'
    2023-07-15','2023-07-21'),(59,35,1,0.2,'2023-08-18','2023-08-24')
    ,(60,36,2,0.05,'2023-09-13','2023-09-19'),(61,45,1,0.1,'2023-04-01','
    2023-04-10'),(62,45,2,0.15,'2023-04-15','2023-04-25'),(63,46,1,0.05,'
    2023-05-01','2023-05-07'),(64,46,2,0.2,'2023-05-15','2023-05-22')
    ,(65,47,1,0.1,'2023-06-01','2023-06-10'),(66,49,1,0.05,'2023-06-15','
    2023-06-20'),(67,49,2,0.1,'2023-07-01','2023-07-15'),(68,50,1,0.25,'
    2023-07-20','2023-07-30'),(69,51,1,0.15,'2023-08-01','2023-08-11')
    ,(70,51,2,0.2,'2023-08-15','2023-08-25'),(71,52,1,0.05,'2023-09-01','
    2023-09-10'),(72,53,2,0.1,'2023-09-15','2023-09-25'),(73,54,1,0.2,'
    2023-10-01','2023-10-07'),(74,55,1,0.1,'2023-10-15','2023-10-22')
    ,(75,56,2,0.15,'2023-11-01','2023-11-15'),(76,57,1,0.25,'2023-11-20',
    '2023-11-30'),(77,58,1,0.05,'2023-12-01','2023-12-10'),(78,59,2,0.2,'
    2023-12-15','2023-12-25'),(79,60,1,0.1,'2024-01-01','2024-01-10')
    ,(80,60,2,0.15,'2024-01-15','2024-01-25');
/*!40000 ALTER TABLE `t_variant_discounts` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-03-30 11:57:02
```

Listing 5: Database with Sample Data

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from fastapi.middleware.cors import CORSMiddleware
import uvicorn

# Import your custom modules here
from llamaSQL_cust_server import nl2sqlquery  # Assuming you have a
    function to handle the queries

app = FastAPI()


app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Allow all origins, you can restrict it based
    on your requirements
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE", "OPTIONS"],  # Allow
   the required HTTP methods
```

```
    allow_headers=["*"],
)
class Query(BaseModel):
    question: str

@app.post("/api/query/")
async def read_query(query: Query):
    try:
        # Use your custom function or class method to process the query
        print(f"Received question: {query.question}")
        response = nl2sqlquery(query.question)
        return {"response": response}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

if __name__ == "__main__":
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

Listing 6: LlamaSQL Server

```
import os
import openai

os.environ["HTTP_PROXY"] = "http://localhost:7890" # huggingface got
    banned by GFW, ToT
os.environ["HTTPS_PROXY"] = "http://localhost:7890"
os.environ["OPENAI_API_KEY"] = "sk-your api" #limited api, hope some
    rich ones can fund this
os.environ["OPENAI_API_BASE"] = "your api base url"
openai.api_key = os.environ["OPENAI_API_KEY"]
openai.base_url = os.environ["OPENAI_API_BASE"]
from IPython.display import Markdown, display

from llama_index.core import Settings, VectorStoreIndex,
    SimpleDirectoryReader
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.llms.replicate import Replicate
from transformers import AutoTokenizer

# Create Database Schema

from sqlalchemy import (
    create_engine,
    MetaData,
    Table,
    Column,
    String,
    Integer,
    select,
)
```

```
from llama_index.core import SQLDatabase
from sqlalchemy import insert
from sqlalchemy import text

# Define SQL Database

engine = create_engine("mysql+pymysql://root:121090155@127.0.0.1:3306/
    comprehensive_supermarket")
metadata_obj = MetaData()
metadata_obj.reflect(bind=engine)

from llama_index.core import SQLDatabase
from llama_index.llms.openai import OpenAI

llm = OpenAI(temperature=0.1, model="gpt-3.5-turbo")
table_list = [
    "t_customers",
    "t_employees",
    "t_expense_reports",
    "t_inventory_reports",
    "t_product_management_records",
    "t_product_variants",
    "t_products",
    "t_reduction_promotions",
    "t_sales_reports",
    "t_supplier_contacts",
    "t_suppliers",
    "t_supply_records",
    "t_transaction_details",
    "t_transaction_records",
    "t_variant_discounts"
]

# Create an instance of SQLDatabase including all tables
sql_database = SQLDatabase(engine, include_tables=table_list)
from sqlalchemy import insert



# Text-to-SQL Retriever

from llama_index.core.retrievers import NLSQLRetriever
from llama_index.core.response.notebook_utils import display_source_node
from llama_index.core.query_engine import RetrieverQueryEngine

def nl2sqlquery(query_str):
    # default retrieval (return_raw=True)
    nl_sql_retriever = NLSQLRetriever(
        sql_database, tables=table_list, return_raw=True
```

```
    )

    results = nl_sql_retriever.retrieve(
        query_str
    )
    for n in results:
        # display_source_node(n)
        print(n)

    query_engine = RetrieverQueryEngine.from_args(nl_sql_retriever)
    response = query_engine.query(
        query_str
    )
    return str(response)

print("listening")
```

Listing 7: LlamaSQL Cust Server