



THE CHINESE UNIVERSITY OF HONG KONG, SHENZHEN

CSC3170 PROJECT DESIGN DOCUMENT

Database System for E-Commerce Retail Company

GROUP 14

Member:

Ge Wentao
Zheng Jin
Dong Jingqi
Song Wenxin
Wang Yuancheng
Zhao Yuzheng

Student Number:

119010080
119020078
119010060
120090625
119010319
119020077

May 14, 2022

Contents

1	Introduction and Motivation	2
2	Database Design and Implementation	2
2.1	Database Design	2
2.1.1	The Entity-Relationship Model	2
2.1.2	The Relational Schema	4
2.1.3	Relational Cardinality	6
2.2	Database Optimization	7
2.2.1	Index	7
2.2.2	Normal Form	7
2.3	Data Implementations	7
2.3.1	Data Generation	7
2.3.2	Database Implementation	8
3	Data Analysis and Data Mining	8
3.1	Basic Functions and Results	8
3.1.1	Supplier	8
3.1.2	Purchase	9
3.1.3	Sell	10
3.1.4	Online Store	10
3.1.5	Customers	10
3.2	Transaction Update Function	11
3.2.1	Stock	11
3.2.2	Sell	11
3.3	Data Mining	12
3.3.1	Customer Retention Rate	12
3.3.2	Customer Analysis	12
3.3.3	Selling Analysis	14
4	Conclusions	14
5	Self-Evaluation	14
6	Contribution	15
7	Appendices	16
7.1	Create Table	16
7.2	Basic SQL Queries	20
7.3	Data Mining SQL Queries	22

1 Introduction and Motivation

Recent years, electronic shopping has developed rapidly. More and more people are joining the e-commerce industry, which has led to huge and complex data in the field that is difficult to manage. Therefore, we propose our project, which aims at implementing a database system for an e-commerce retail company. Our goal is to build a system that is capable for data organization, data storage, and data analysis. Even people who are not computer literate and do not understand data analysis can make decisions well with our database.

2 Database Design and Implementation

In this section, we will focus on illustrating the requirement and specification of the database, designing the Entity-Relationship Model, reducing ER diagram into relational schemas, and set our fundamental assumptions.

2.1 Database Design

2.1.1 The Entity-Relationship Model

- **E-R Diagram:** In the E-R model, there are 7 categories of entities including Suppliers, Purchase, Inventories, Orders, Customers, Online Store and Locations. In total, there are 12 entities and 23 relationships. The E-R diagram of the model is shown below:

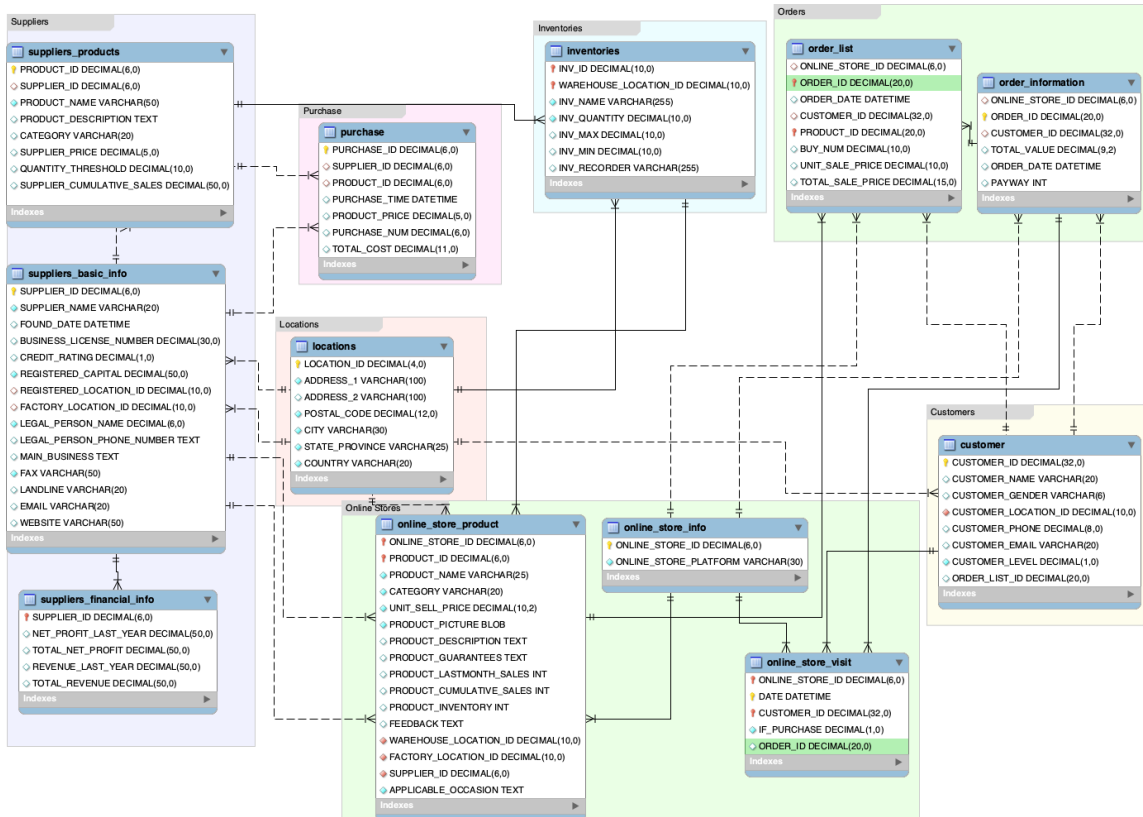


Figure 1: E-R Diagram

- **Entity sets:** The database stores necessary information about entities and their relationships. Entities and their descriptions are listed in the table below:

Table 1: Entities in the database

Category	Entity	Description
Suppliers	suppliers_basic_info	Contain the basic information of all suppliers
	suppliers_financial_info	Contain the financial information of all suppliers
	suppliers_products	Contain the products that each supplier supplies
Purchase	purchase	Contain the information about purchasing product from suppliers
Inventories	inventories	Contain the inventories of products that each supplier possess
Locations	locations	Contain the location information
Online Store	online_store_info	Contain the information about the online store
	online_store_product	Contain the products and product information that are in the online store
	online_store_visit	Contain the customer access (to the store) information
Customers	customers	Contain the customer information
Orders	order_information	Contain the detailed information about orders
	order_list	Contain the list of orders in each online store

2.1.2 The Relational Schema

In this section, we reduce the E-R diagram into relational schema to help us sort out the relationships between different entities and check the dependencies. The relational schema in Third Normal Form for our database system is listed below:

- `supplier_financial_info`(supplier_id, net_profit_last_year, total_net_profit, revenue_last_year, total_revenue)
- `supplier_basic_info`(supplier_id, supplier_name, found_date, business_license_num, credit_rating, registered_capital, registered_location_id, factory_location_id, legal_person_id, legal_person_phone_number, main_business, fax, landline, email, website)
- `supplier_product`(product_id, supplier_id, product_name, product_description, category, supplier_price, quantity_threshold, supplier_cumulative_sales)
- `purchase`(purchase_id, supplier_id, product_id, purchase_time, product_price, purchase_num, total_cost)
- `locations`(location_id, address_1, address_2, postal_code, city, state_province, country)
- `inventories`(inv_id, warehouse_location_id, inv_name, inv_quantity, inv_max, inv_min, inv_recorder)
- `online_store_products`(online_store_id, product_id, product_name, category, feedback, unit_sale_price, product_picture, product_description, product_guarantees, product_last_month_sales, product_cumulative_sales, product_inventory, applicable_occasion, warehouse_location_id, factory_location_id, supplier_id)
- `online_store_info`(online_store_id, online_store_platform)
- `online_store_visit`(online_store_id, customer_id, date, if_purchase, order_id)
- `customer`(customer_id, customer_name, customer_gender, customer_location_id, customer_phone, customer_email, customer_level, order_list_id)
- `order_info`(order_id, total_value, order_date, payway, online_store_id, customer_id)
- `order_list`(order_id, product_id, order_date, buy_num, unit_sale_price, total_sale_price, online_store_id, customer_id)

The figure below shows the relational schema with foreign key referencing.

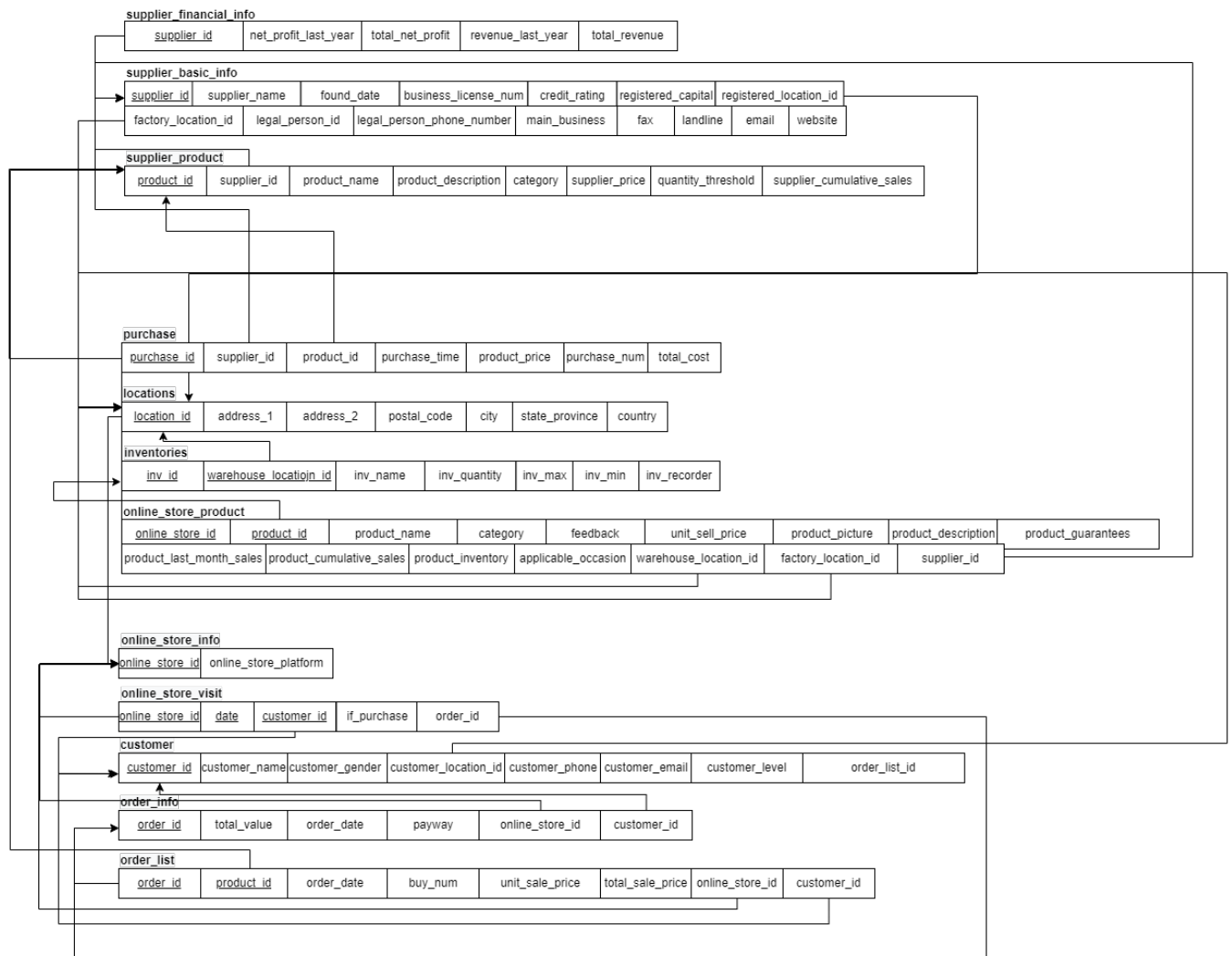


Figure 2: Relational Schema Showing Foreign Key Referencing

2.1.3 Relational Cardinality

After getting the relational schema, we sort out the relationship between entities as well as the relationship cardinalities. The relationship cardinalities are shown in the table below.

Table 2: Relationship Cardinalities

Entity Involved	Cardinality Constraint	Description
purchase suppliers_products	One to one	A purchase entity instance contains information about buying one specific product from suppliers.
purchase suppliers_basic_info	One to one	A purchase entity instance contains information about buying products from which supplier.
inventories suppliers_products	One to one	Each product has its corresponding inventory.
inventories locations	One to one	Each inventory has its corresponding warehouse location.
suppliers_basic_info locations	One to one	Each supplier has its corresponding registered location and factory location.
customers orders_information	One to many	A customer can place many orders.
order_list order_information	Many to many	There can be several orders in one order list. An order can appear in different order lists.
customer online_store_visit	One to many	A customer can visit many online stores.
online_store_info online_store_products	Many to many	An online store can contains many products. One product can be sold in multiple online stores.

2.2 Database Optimization

2.2.1 Index

Indexing is a powerful structure in MySQL which can make searching more effective from common queries. Indexes are usually implemented in two ways, using B tree and using hashing. Using B tree can reduce the searching time from $O(n)$ to $O(\log n)$ and using hashing can reduce searching time from $O(n)$ to $O(1)$. However, there are some problems with using hashing, it may increase the search time due to the existence of hash collisions, and if we want to have almost no collisions, we need more disk storage to save the values. Therefore, we often chose using B tree. It is worth noting that although indexing makes search more efficient, it will lead to increased data modification or addition and deletion time, so we often add indexes to data that is frequently searched and infrequently changed. In details, we create indexes to the following attributes:

- All primary keys are indexed by default.
- `supplier_name`, `purchase_time`, `product_name`, `customer_name`, and `order_date`.

2.2.2 Normal Form

In order to approach for relational databases which uses normalizing principles to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management, most schemas of our database are in third norm form (3NF).

2.3 Data Implementations

We took 2 steps to implement our database design:

1. Generate data according to our design.
2. Create the database system and import our data.

We will illustrate each step in this section.

2.3.1 Data Generation

After designing database structure, we generate data to implement our design. Our data can be divided into 7 parts: Inventories, Locations, Customers, Suppliers, Purchases, Orders, and Online Store.

As for Inventories, we collect the inventory names from Amazon website. We assume each inventory has a minimum quantity of 0, and we assign the current quantity and maximum quantity appropriate positive integer numbers. For the foreign key `Location_ID`, each Inventory is assigned to an appropriate storage location.

As for locations, customers, and suppliers, we find these data on dataworld website (see appendix), which is a public data website.

For suppliers, we also need their financial information and the products they supply to analyze what products to import from which supplier. These financial data such as revenue

and suppliers' products data such supplier price are assigned to appropriate numbers.

Purchases and Orders are tables that record daily transactions. Each purchase transitions are generated by appropriately combining supplier information and inventory information. The purchase number is reasonably assigned and total cost is calculated by $\text{price} \times \text{purchase number}$. For orders, we assume each customer can buy more than one product in an order. Order information is generated by combining customer and online store. Order list table records what product a customer buy in each order, which is a combination of inventories and order information.

As for online store, we assume that we have three online store on Taobao, Tiktok, and Jingdong, which is represented by 1, 2, and 3 in online store ID respectively. We generate appropriate data for online store visit and online store product to simulate who visit the online store, whether he or she buy anything and what products are sold in each store.

2.3.2 Database Implementation

After collecting and generating data, we firstly standardize their format. Secondly, we add primal key and foreign key constraints to each table. primal keys are computer generated ascending integers. For example, 'INV_ID' is positive integers start from 1. Foreign keys are also appropriately assigned to each table. For example, 'Location_ID' is the foreign key of inventoried, customers and suppliers. Each entity and relation are implemented by Workbench and VSC.

3 Data Analysis and Data Mining

Utilizing the database system for e-commerce retail company and combining with machine learning algorithms, we can perform some basic data analysis and data mining techniques. Based on the generated result, we can have further suggestions towards the company's operation.

3.1 Basic Functions and Results

This project includes 8 queries to gain information from the database. They can be divided by different subjects in the database, which are Supplier, Customer and Retailing Company. Retailing Company is further divided in to Online Store, Sell and Purchase for emphasis. Detailed query statements are included in Appendix 8.1

3.1.1 Supplier

- **Profit Revenue**

Search and compare the profit revenue for different suppliers. The result is first ordered by the net profit of each supplier last year and then ordered by their total net profit. The amount of net profit last year can demonstrate their recent performance, while the total net balance can show their strong management ability. The result can be

used to decide the proportion of goods purchased from different suppliers. The result is shown in Figure 2 (only part of the result is shown).

SUPPLIER_ID	NET_PROFIT_LAST_YEAR	TOTAL_NET_PROFIT
10	878756	3878756
9	3454576	83454576
6	4324324	54324324
4	4342341	44342341
7	4554554	64554554
20	5452663	35452663
15	6245692	62456921
11	6567665	56567665
12	7675768	67675768

Figure 3: Profit Revenue

- **The minimum purchasing cost for the same product**

Search and find the minimum purchasing cost for the same product from different suppliers. This online retailing company will purchase the same products from different suppliers. And these products may have different price. The minimum purchase price is essential when deciding the selling price. Thus the result can be used as a reference when deciding the selling price. The result is shown in Figure 3 (only part of the result is shown).

PRODUCT_ID	SUPPLIER_ID	PRODUCT_NAME	CATEGORY	MINIMUM_PRICE
1	1	ANCO MILK	DRINK	3
2	1	ASIA ORANGE	FRUIT	4
3	6	BOY LONDON SPRING SHORT	CLOTH	99
4	7	SONY PS5	ENTERTAINMENT	800
5	8	NICE SHAMPOO	DAILY USE	20
4	7	SONY PS5	ENTERTAINMENT	800
5	8	NICE SHAMPOO	DAILY USE	20
11	17	MACBOOK PRO M1 32+1TB	ELECTRONIC	2999

Figure 4: The Minimum Purchasing Cost

3.1.2 Purchase

- **The supplier that this company order the most from**
- **The product that this company order the most**

The results can serve as reference for purchasing department. It can try to contacting the supplier that the company order the most for cooperation in order to reduce the purchase cost, and eventually boost profits for the company. The results are shown in Figure 4 and 5, respectively (only parts of the results are shown).

SUPPLIER_ID	SUM(TOTAL_COST)
1	64
2	114.6
3	198
5	200

Figure 5: The Most Ordered Supplier

PRODUCT_ID	SUM(TOTAL_COST)
17	16.5
16	180
20	180

Figure 6: The Most Ordered Product

3.1.3 Sell

- **The selling records for different months**

Find all the selling records for different months. The result is grouped by months. The selling records can be used to analyze the most popular items for each month or different seasons. The retailing department can use the result to adjust the purchase quantity for some products from suppliers in different month. The result is shown in Figure 6 (only part of the result is shown).

PRODUCT_ID	ORDER_MONTH	SALES
1	3	45
2	3	2599
1	4	55
2	4	5198

Figure 7: The Selling Records for Different Months

3.1.4 Online Store

- **The average conversion rate**

Calculate the average conversion rate for three online stores of this company on different dates. This company owns three online stores on different retailing platform, such as Tao Bao, Jing Dong and Pin Duo Duo. When a customer visit any of the stores, they visit will be recorded. They may place an order or not. The conversion rate is ratio of the number of records of transaction over the number of visit records. The result can be help the manager to decide whether the store interface is attractive enough or the display order is reasonable. The result is shown in Figure 7 (only part of the result is shown).

DATE	Daily_Conversion_Rate
2022/3/17	33.30%
2022/3/18	50%
2022/3/19	100.00%
2022/3/20	66.70%

Figure 8: The Average Conversion Rate

3.1.5 Customers

- **The detailed information of order**

Find the information of order for a specific customer by order id. Online store usually needs to deal with after sales problems. The information of order can help the customer server know better of the products the customer has bought, so that can provide better service for them. The result is shown in Figure 8 (only part of the result is shown).

ORDER_ID	ONLINE_STORE_ID	CUSTOMER_ID	PRODUCT_ID	BUY_NUM	UNIT_SALE_PRICE	TOTAL_SALE_PRICE
1	1	10001	1	2	5	10
1	1	10001	2	2	7	14
1	1	10001	3	1	499	499

Figure 9: The Order Information

- **The total amount of consumption for different customers**

Find the total amount of consumption for different customers. This company will offer different account for customers with different grade of membership. The total consumption can be used to decide the grade of membership for each customer. The result is shown in Figure 9 (only part of the result is shown).

CUSTOMER_ID	TOTAL_VALUE
1	523
2	2599
3	69

Figure 10: The Total Amount of Consumption

3.2 Transaction Update Function

Daily transactions such as stock and sell are implemented by transaction update functions. This section will show how transaction update function works to record daily transactions.

3.2.1 Stock

A stock transaction will make change on table purchase and inventories. After starting a transaction, we first insert a purchase information in table purchase, then according to the quantity of purchase, we update table inventories, adding the amount of inventory purchased. If the amount of that inventory is larger than the maximum quantity, then abort the transaction. Otherwise, commit the transaction.

3.2.2 Sell

A sell transaction will make change on table order_information, order_list, and inventories. A sell transaction starts at inserting a new order in order_information. We assume that a customer can buy different kinds of products in one order. So, we may insert more than one data in order_list. Finally, the insertion on order_list can make change on the quantity of inventories. If the amount of any inventory is less than zero, then abort the transaction. Otherwise, commit the transaction.

3.3 Data Mining

3.3.1 Customer Retention Rate

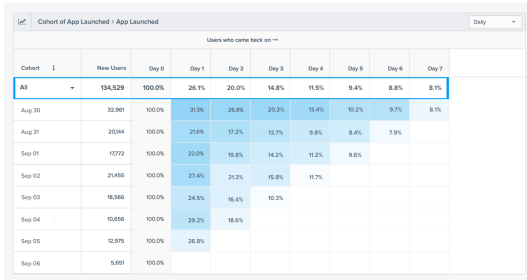
Improve the customer retention rate is vital for the company's development. It is widely recognized that it costs less to keep an existing customer than to require a new one.

Day-N new customer retention rate is the percentage of customers who visit the website again N day after they registered an account when they came to the website for the first time. The formula is as follows.

$$\text{Day} - N \text{ retention rate} = \frac{R_{t_N}(U_{t_0})}{U_{t_0}}$$

Here, U_{t_0} is the number of new customers at day t_0 . $R_{t_N}(U_{t_0})$ represents the number of new customers registering at day t_0 and visiting the website at day t_N .

Our query is attached to the Appendix and it will show the customer retention frequency and rate in day-1, day-2, day-3, day-4, day-5, day-6, day-7, day-15, day-30 as the sample customer retention rate output. According to the output, we can roughly observe that the customer retention rate decreases with the time interval increases, which forms an convex curve. Also, with the retention rate data, we can monitor the customer activity and the effect of our operation strategies is reflected. Based on the customer retention rate, we can make some further justification towards our operation strategies to improve it and develop the company's product.



Cohort	New Users	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
All	134,529	100.0%	26.1%	20.0%	14.8%	11.5%	9.4%	8.8%	8.3%
Aug 30	32,981	100.0%	31.3%	26.8%	20.2%	15.4%	10.2%	9.7%	8.7%
Aug 31	20,044	100.0%	21.8%	17.2%	13.7%	9.8%	8.4%	7.9%	
Sep 01	12,772	100.0%	22.0%	19.8%	14.2%	11.2%	9.6%		
Sep 02	21,455	100.0%	27.4%	21.3%	15.8%	11.7%			
Sep 03	18,566	100.0%	24.5%	16.4%	10.3%				
Sep 04	10,858	100.0%	29.2%	18.6%					
Sep 05	12,875	100.0%	26.8%						
Sep 06	8,891	100.0%							

Figure 11: Sample Customer Retention Output

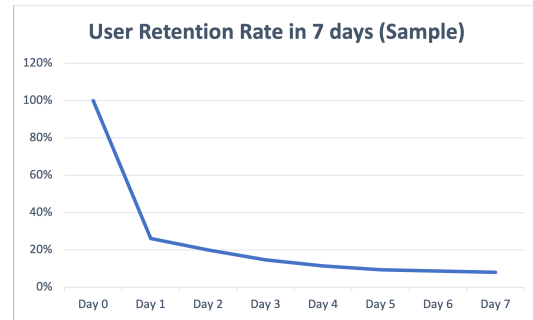


Figure 12: Customer Retention Rate Trend

From the sample output, we can draw a conclusion that the customer retention rate decreases when time interval N increases. The customer retention rate is almost below 30% for day 1 and for day 7, the rate is below 10%.

3.3.2 Customer Analysis

For the customer analysis part, we will focus on analyzing the difference in consumption habit of different products between different gender, different region. By conducting the customer analysis, we can obtain the consumption habits of customers from different regions and genders. On this basis, we can recommend product to customers with different characteristics accurately and help improve the purchase rate and customer retention rate.

Firstly, we analyze the consumption habit of different genders. we use the sample query to get the total consumption amount and the average consumption amount for the male and female customer. The output is shown in the below image. From the sample output, we can observe that the majority of our customers are female who takes up 67% percentage and female customers always have larger amount in total consumption and average consumption. Thus, the company can promote more products that cater to female customers' preferences and get further development.

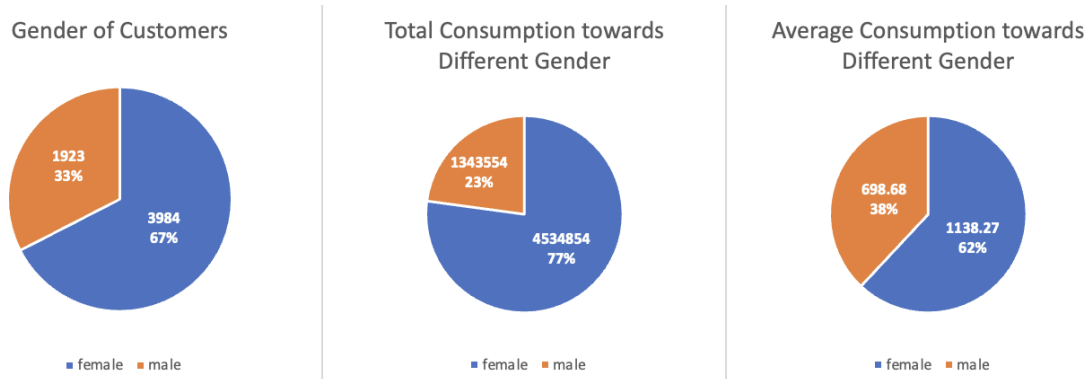


Figure 13: Consumption of Customers of Different Genders

Secondly, we analyze the consumption habit of customers from different states. We use the sample query to analyze the distribution of region of customers and the amount of customers' total consumption and average consumption from different states. According to the sample output, we found that California State has the largest portion of customers and the largest total consumption amount. However, Colorado State has the largest average consumption amount. Thus, the company can develop a new strategy that focus on holding some events like selling Limited edition items and the customer party in California State to retain and activate customers. For Colorado State, we can increase the marketing of the company's product and attract more potential customers.

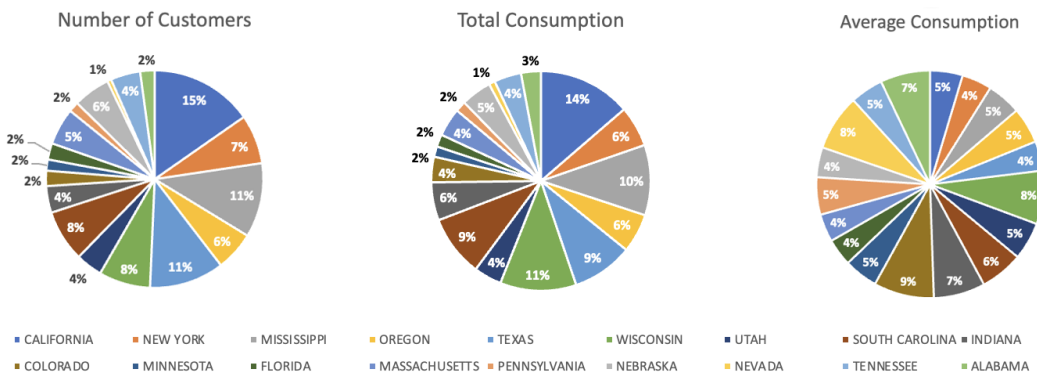


Figure 14: Consumption of Customers of Different States

3.3.3 Selling Analysis

we track and analyze the selling amount of products. We use the sample query to get the time period where the product has continuous increasing in selling amount. This can help check the peak season for products and Conduct the marketing of different products according to the season. The sample output is shown as follows.

ORDER_DATE	PRODUCT_ID	AMOUNT
2019/3/24	1	30
2019/3/25	1	40
2019/3/26	1	42
2019/3/2	2	10
2019/3/3	2	13
2019/3/4	2	15
2019/3/5	2	20
2019/3/4	3	2
2019/3/5	3	4

Figure 15: Sample Query Output

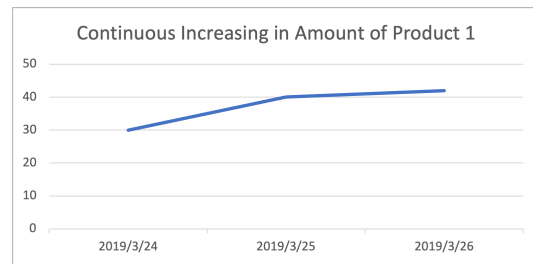


Figure 16: Visualization of the part of the result

4 Conclusions

In this project, we considered e-commerce companies that connected with supplier and customers with strong supply chain as well as sell their products at different platforms like Tiktok and Taobao and companies require a powerful database system to manage its supply chain, sales and customer information data. After analyzing the company's demands, we designed a Supply chain, Sales and customer information data Management System to ensure efficient operations of the company.

We recognized suppliers, customers, online stores, orders, locations, products as main entities and studied the relation between them. Then, we designed the ER diagram and tried to combine relations with entities to improve query efficiency and lower the data redundancy level. We also reduce the refined ER diagram to the relational schemas and verified that they are in the Boyce Codd Normal Form.

Besides, we tried to bring our design into reality by generating data, establishing a real database, importing them into the database. To demonstrate our database system in a more straightforward way, we created some sample queries. In addition, we also visualized certain situations where our database can be used for data mining purposes. The abovementioned sample queries and data mining potentials have proved that our database system can satisfy not only the company's daily operation but also further demands in data analysis.

5 Self-Evaluation

In this project, we learned to think from both the company's view and from a database engineers' view. It helps us better understand the actual implementation of a database and

the real need for a database. From deciding on which topic to do, everyone in our group showed anticipation in this project. The discussion and job splitting are efficient. The database we design is simple but complete.

Limited by time and data privacy, it is a well-developed e-commerce data management system consists of considerable design, thought-provoking ideas, and theoretical support for vaccine distribution and implementation strategies.

6 Contribution


Everyone contributes equally to the project.

7 Appendices

7.1 Create Table

```
DROP SCHEMA IF EXISTS `project` ;  
CREATE SCHEMA IF NOT EXISTS `project` DEFAULT CHARACTER SET utf8 ;  
USE `project` ;
```

Figure 17: Default Setting for Creating Table



```
CREATE TABLE `project`.`locations` (  
    LOCATION_ID DECIMAL(4,0),  
    ADDRESS_1 VARCHAR(100) NOT NULL,  
    ADDRESS_2 VARCHAR(100),  
    POSTAL_CODE DECIMAL(12,0) NOT NULL,  
    CITY VARCHAR(30) NOT NULL,  
    STATE_PROVINCE VARCHAR(25) NOT NULL,  
    COUNTRY VARCHAR(20) NOT NULL,  
    PRIMARY KEY (LOCATION_ID)  
);
```

Figure 18: Create Table for Locations



```
CREATE TABLE `project`.`purchase` (  
    PURCHASE_ID DECIMAL(6, 0),  
    SUPPLIER_ID DECIMAL(6, 0),  
    PRODUCT_ID DECIMAL(6, 0),  
    PURCHASE_TIME DATETIME,  
    PRODUCT_PRICE DECIMAL(5, 0),  
    PURCHASE_NUM DECIMAL(6, 0),  
    TOTAL_COST DECIMAL(11, 0),  
    PRIMARY KEY (PURCHASE_ID),  
    FOREIGN KEY (SUPPLIER_ID) REFERENCES `project`.`suppliers_basic_info` (SUPPLIER_ID),  
    FOREIGN KEY (PRODUCT_ID) REFERENCES `project`.`suppliers_products` (PRODUCT_ID)  
);
```

Figure 19: Create Table for Purchase

```

CREATE TABLE `project`.`inventories` (
  INV_ID DECIMAL(10,0),
  WAREHOUSE_LOCATION_ID DECIMAL(10,0),
  INV_NAME VARCHAR(255) NOT NULL,
  INV_QUANTITY DECIMAL(10,0) NOT NULL DEFAULT 0,
  INV_MAX DECIMAL(10,0),
  INV_MIN DECIMAL(10,0) DEFAULT 0,
  INV_RECORDER VARCHAR(255),
  PRIMARY KEY (INV_ID, WAREHOUSE_LOCATION_ID),
  FOREIGN KEY (INV_ID) REFERENCES `project`.`suppliers_products` (PRODUCT_ID),
  FOREIGN KEY (WAREHOUSE_LOCATION_ID) REFERENCES `project`.`locations` (LOCATION_ID)
);

```

Figure 20: Create Table for Inventories

```

CREATE TABLE `project`.`suppliers_basic_info` (
  SUPPLIER_ID DECIMAL(6,0),
  SUPPLIER_NAME VARCHAR(20) NOT NULL,
  FOUND_DATE DATETIME,
  BUSINESS_LICENSE_NUMBER DECIMAL(30,0),
  CREDIT_RATING DECIMAL(1,0),
  REGISTERED_CAPITAL DECIMAL(50,0) NOT NULL,
  REGISTERED_LOCATION_ID DECIMAL(10,0),
  FACTORY_LOCATION_ID DECIMAL(10,0),
  LEGAL_PERSON_NAME DECIMAL(6,0) NOT NULL,
  LEGAL_PERSON_PHONE_NUMBER TEXT,
  MAIN_BUSINESS TEXT,
  FAX VARCHAR(50) NOT NULL,
  LANDLINE VARCHAR(20),
  EMAIL VARCHAR(20),
  WEBSITE VARCHAR(50),
  PRIMARY KEY (SUPPLIER_ID),
  FOREIGN KEY (REGISTERED_LOCATION_ID) REFERENCES `project`.`locations` (LOCATION_ID),
  FOREIGN KEY (FACTORY_LOCATION_ID) REFERENCES `project`.`locations` (LOCATION_ID)
);

```

Figure 21: Create Table for Supplier Basic Information

```

CREATE TABLE `project`.`suppliers_financial_info` (
  SUPPLIER_ID DECIMAL(6,0),
  NET_PROFIT_LAST_YEAR DECIMAL(50,0),
  TOTAL_NET_PROFIT DECIMAL(50,0),
  REVENUE_LAST_YEAR DECIMAL(50,0),
  TOTAL_REVENUE DECIMAL(50,0),
  PRIMARY KEY (SUPPLIER_ID),
  FOREIGN KEY (SUPPLIER_ID) REFERENCES `project`.`suppliers_basic_info` (SUPPLIER_ID)
);

```

Figure 22: Create Table for Supplier Financial Information

```
CREATE TABLE `project`.`suppliers_products` (  
    PRODUCT_ID DECIMAL(6,0),  
    SUPPLIER_ID DECIMAL(6,0),  
    PRODUCT_NAME VARCHAR(50) NOT NULL,  
    PRODUCT_DESCRIPTION TEXT,  
    CATEGORY VARCHAR(20),  
    SUPPLIER_PRICE DECIMAL(5,0),  
    QUANTITY_THRESHOLD DECIMAL(10,0),  
    SUPPLIER_CUMULATIVE_SALES DECIMAL(50,0),  
    PRIMARY KEY (PRODUCT_ID),  
    FOREIGN KEY (SUPPLIER_ID) REFERENCES `project`.`suppliers_basic_info` (SUPPLIER_ID)  
);
```

Figure 23: Create Table for Products Provided by Suppliers

```
CREATE TABLE `project`.`customer` (  
    CUSTOMER_ID DECIMAL(32,0),  
    CUSTOMER_NAME VARCHAR(20),  
    CUSTOMER_GENDER VARCHAR(6),  
    CUSTOMER_LOCATION_ID DECIMAL(10,0) NOT NULL,  
    CUSTOMER_PHONE DECIMAL(8,0),  
    CUSTOMER_EMAIL VARCHAR(20),  
    CUSTOMER_LEVEL DECIMAL(1,0) NOT NULL,  
    ORDER_LIST_ID DECIMAL(20,0),  
    PRIMARY KEY(CUSTOMER_ID),  
    FOREIGN KEY(CUSTOMER_LOCATION_ID) REFERENCES `project`.`locations` (LOCATION_ID)  
);
```

Figure 24: Create Table for Customers

```
CREATE TABLE `project`.`online_store_info` (  
    ONLINE_STORE_ID DECIMAL(6,0),  
    ONLINE_STORE_PLATFORM VARCHAR(30) NOT NULL,  
    PRIMARY KEY (ONLINE_STORE_ID)  
);
```

Figure 25: Create Table for Online Stores

```

CREATE TABLE `project`.`online_store_product` (
  ONLINE_STORE_ID DECIMAL(6,0),
  PRODUCT_ID decimal(6,0) NOT NULL,
  PRODUCT_NAME varchar(25) NOT NULL,
  CATEGORY varchar(20) NOT NULL,
  UNIT_SELL_PRICE decimal(10,2) NOT NULL,
  PRODUCT_PICTURE BLOB NOT NULL,
  PRODUCT_DESCRIPTION TEXT,
  PRODUCT_GUARANTEES TEXT,
  PRODUCT_LASTMONTH_SALES INT,
  PRODUCT_CUMULATIVE_SALES INT,
  PRODUCT_INVENTORY INT,
  FEEDBACK TEXT,
  WAREHOUSE_LOCATION_ID DECIMAL(10,0) NOT NULL,
  FACTORY_LOCATION_ID DECIMAL(10,0) NOT NULL,
  SUPPLIER_ID DECIMAL(6,0) NOT NULL,
  APPLICABLE_OCCASION TEXT NOT NULL,
  PRIMARY KEY (ONLINE_STORE_ID, PRODUCT_ID),
  FOREIGN KEY (ONLINE_STORE_ID) REFERENCES `project`.`online_store_info` (ONLINE_STORE_ID),
  FOREIGN KEY (PRODUCT_ID) REFERENCES `project`.`inventories` (INV_ID),
  FOREIGN KEY (WAREHOUSE_LOCATION_ID) REFERENCES `project`.`locations` (LOCATION_ID),
  FOREIGN KEY (FACTORY_LOCATION_ID) REFERENCES `project`.`suppliers_basic_info` (FACTORY_LOCATION_ID),
  FOREIGN KEY (SUPPLIER_ID) REFERENCES `project`.`suppliers_basic_info` (SUPPLIER_ID)
);

```

Figure 26: Create Table for Products of Online Stores

```

CREATE TABLE `project`.`online_store_visit` (
  ONLINE_STORE_ID DECIMAL(6,0),
  `DATE` DATETIME,
  CUSTOMER_ID DECIMAL(32,0),
  IF_PURCHASE DECIMAL(1,0) NOT NULL,
  ORDER_ID DECIMAL(20,0) DEFAULT NULL,
  PRIMARY KEY (ONLINE_STORE_ID, `DATE`, CUSTOMER_ID),
  FOREIGN KEY (ONLINE_STORE_ID) REFERENCES `project`.`online_store_info` (ONLINE_STORE_ID),
  FOREIGN KEY (CUSTOMER_ID) REFERENCES `project`.`customer` (CUSTOMER_ID),
  FOREIGN KEY (ORDER_ID) REFERENCES `project`.`order_information` (ORDER_ID)
);

```

Figure 27: Create Table for Visiting Information of Online Stores

```

CREATE TABLE `project`.`order_list` (
  ONLINE_STORE_ID DECIMAL(6,0),
  ORDER_ID DECIMAL(20,0),
  ORDER_DATE DATETIME,
  CUSTOMER_ID DECIMAL(32,0),
  PRODUCT_ID DECIMAL(20,0),
  BUY_NUM DECIMAL(10,0),
  UNIT_SALE_PRICE DECIMAL(10,0),
  TOTAL_SALE_PRICE DECIMAL(15,0),
  PRIMARY KEY (ORDER_ID, PRODUCT_ID),
  FOREIGN KEY (ORDER_ID) REFERENCES `project`.`order_information` (ORDER_ID),
  FOREIGN KEY (CUSTOMER_ID) REFERENCES `project`.`customer` (CUSTOMER_ID),
  FOREIGN KEY (PRODUCT_ID) REFERENCES `project`.`online_store_product` (PRODUCT_ID),
  FOREIGN KEY (ONLINE_STORE_ID) REFERENCES `project`.`online_store_info` (ONLINE_STORE_ID)
);

```

Figure 28: Create Table for the Order List

```

CREATE TABLE `project`.`order_information` (
  ONLINE_STORE_ID DECIMAL(6,0),
  ORDER_ID DECIMAL(20,0),
  CUSTOMER_ID DECIMAL(32,0),
  TOTAL_VALUE DECIMAL(9,2),
  ORDER_DATE DATETIME DEFAULT NULL,
  PAYWAY INT,
  PRIMARY KEY (ORDER_ID),
  FOREIGN KEY (ONLINE_STORE_ID) REFERENCES `project`.`online_store_info` (ONLINE_STORE_ID),
  FOREIGN KEY (CUSTOMER_ID) REFERENCES `project`.`customer` (CUSTOMER_ID)
);

```

Figure 29: Create Table for the Order Specific Information

```

CREATE INDEX SUPPLIER_NAME_INDEX ON `project`.`suppliers_basic_info` (SUPPLIER_NAME);
CREATE INDEX PURCHASE_TIME_INDEX ON `project`.`purchase` (PURCHASE_TIME);
CREATE INDEX PRODUCT_NAME_INDEX ON `project`.`suppliers_products` (PRODUCT_NAME);
CREATE INDEX CUSTOMER_NAME_INDEX ON `project`.`customer` (CUSTOMER_NAME);
CREATE INDEX ORDER_DATE_INDEX ON `project`.`order_list` (ORDER_DATE);

```

Figure 30: Create Index

7.2 Basic SQL Queries

```

SELECT SUPPLIER_ID, NET_PROFIT_LAST_YEAR, TOTAL_NET_PROFIT
FROM `suppliers_financial_info`
ORDER BY NET_PROFIT_LAST_YEAR, TOTAL_NET_PROFIT;

```

Figure 31: Query for finding the profit revenue

```

SELECT
  `DATE`, CONCAT(ROUND(SUM(IF_PURCHASE)/COUNT(CUSTOMER_ID), 4), '%') as Daily_Conversion_Rate
FROM
  `online_store_visit`
GROUP BY `DATE`;

```

Figure 32: Query for getting the average conversion rate

```

SELECT
  *
FROM
  (
    (SELECT PRODUCT_ID, SUPPLIER_ID, PRODUCT_NAME, CATEGORY, MIN(SUPPLIER_PRICE) AS MINIMUM_PRICE
     FROM `suppliers_products` GROUP BY PRODUCT_ID) a
    LEFT JOIN (SELECT * FROM `suppliers_basic_info`) b
    ON a.SUPPLIER_ID = b.SUPPLIER_ID
  )
GROUP BY PRODUCT_ID;

```

Figure 33: Query for finding the minimum purchasing cost for the same product

Query for finding the selling records for different months.

```
select PRODUCT_ID, ORDER_MONTH, a 'SALES' from
(select PRODUCT_ID, MONTH(ORDER_DATE) AS ORDER_MONTH, TOTAL_SALE_PRICE a, 0 e
from order_list
union
select 'MONTH_SUM' PRODUCT_ID ,MONTH(ORDER_DATE) AS ORDER_MONTH, sum(TOTAL_SALE_PRICE) a ,2 e
from order_list
group by MONTH(ORDER_DATE)) lb
order by ORDER_MONTH, e;
```

Figure 34: Query for finding the minimum purchasing cost for the same product

```
-- FIND THE DETAILED INFORMATION FOR A SPECIFIC ORDER(CAN BE USED TO FIND THE ORDER INFORMATION)
SELECT ORDER_ID, ONLINE_STORE_ID, PRODUCT_ID, BUY_NUM, UNIT_SALE_PRICE, TOTAL_SALE_PRICE
FROM order_list
WHERE ORDER_ID = "ORDER_ID";
```

Figure 35: Query for getting the detailed information of order

```
-- FIND THE TOTAL CONSUMPTION FOR A CUSTOMER
SELECT SUM(TOTAL_VALUE)
FROM order_information
WHERE CUSTOMER_ID = "CUSTOMER_ID";
```

Figure 36: Query for the getting total amount of consumption for different customers

```
SELECT SUPPLIER_ID, SUM(TOTAL_COST)
FROM `purchase`
GROUP BY SUPPLIER_ID
ORDER BY SUM(TOTAL_COST);
```

Figure 37: Query for the finding supplier that this company order the most from

```
SELECT PRODUCT_ID, SUM(TOTAL_COST)
FROM `purchase`
GROUP BY PRODUCT_ID
ORDER BY SUM(TOTAL_COST);
```

Figure 38: Query for the finding product that this company order the most

7.3 Data Mining SQL Queries

```
USE project;

CREATE VIEW user_remain_view AS SELECT
a.ORDER_DATE,
count( DISTINCT a.CUSTOMER_ID ) AS user_count,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 1, a.CUSTOMER_ID, NULL ) ) ) AS remain1,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 2, a.CUSTOMER_ID, NULL ) ) ) AS remain2,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 3, a.CUSTOMER_ID, NULL ) ) ) AS remain3,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 4, a.CUSTOMER_ID, NULL ) ) ) AS remain4,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 5, a.CUSTOMER_ID, NULL ) ) ) AS remain5,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 6, a.CUSTOMER_ID, NULL ) ) ) AS remain6,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 7, a.CUSTOMER_ID, NULL ) ) ) AS remain7,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 15, a.CUSTOMER_ID, NULL ) ) ) AS remain15,
count( DISTINCT ( IF ( DATEDIFF( b.ORDER_DATE, a.ORDER_DATE ) = 30, a.CUSTOMER_ID, NULL ) ) ) AS remain30
FROM
( SELECT CUSTOMER_ID, ORDER_DATE FROM `order_information` GROUP BY CUSTOMER_ID, ORDER_DATE ) a
LEFT JOIN ( SELECT CUSTOMER_ID, ORDER_DATE FROM `order_information` GROUP BY CUSTOMER_ID, ORDER_DATE ) b
ON a.CUSTOMER_ID = b.CUSTOMER_ID
WHERE
b.ORDER_DATE >= a.ORDER_DATE
GROUP BY
a.ORDER_DATE;
```

Figure 39: Customer Retention Frequency

```
SELECT
ORDER_DATE,
user_count,
concat( round( remain1 / user_count * 100, 2 ), '%' ) AS day1,
concat( round( remain2 / user_count * 100, 2 ), '%' ) AS day2,
concat( round( remain3 / user_count * 100, 2 ), '%' ) AS day3,
concat( round( remain4 / user_count * 100, 2 ), '%' ) AS day4,
concat( round( remain5 / user_count * 100, 2 ), '%' ) AS day5,
concat( round( remain6 / user_count * 100, 2 ), '%' ) AS day6,
concat( round( remain7 / user_count * 100, 2 ), '%' ) AS day7,
concat( round( remain15 / user_count * 100, 2 ), '%' ) AS day15,
concat( round( remain30 / user_count * 100, 2 ), '%' ) AS day30
FROM
user_remain_view;
```

Figure 40: Customer Frequency Rate

```
USE project;

SELECT
COUNT(*) as GENDER_NUM, SUM(TOTAL_VALUE) as TOTAL_CONSUMPTION, AVG(TOTAL_VALUE) as AVERAGE_CONSUMPTION
FROM
`customer` t1 LEFT JOIN `order_information` t2 ON t1.ORDER_LIST_ID = t2.ORDER_ID
GROUP BY
CUSTOMER_GENDER;

SELECT
COUNT(*) as STATE_NUM, SUM(TOTAL_VALUE) as TOTAL_CONSUMPTION, AVG(TOTAL_VALUE) as AVERAGE_CONSUMPTION
FROM
( `customer` t1 LEFT JOIN `order_information` t2 ON t1.ORDER_LIST_ID = t2.ORDER_ID )
LEFT JOIN `locations` t3 ON t1.CUSTOMER_LOCATION_ID = t3.LOCATION_ID
GROUP BY STATE_PROVINCE;
```

Figure 41: Customer Analysis

```
• USE project;

• SELECT
PRODUCT_ID, ORDER_DATE as ORDER_DATE, sum(BUY_NUM) as BUY_NUM
FROM
order_list
GROUP BY
PRODUCT_ID, ORDER_DATE;

• SELECT a.ORDER_DATE, a.PRODUCT_ID, a.BUY_NUM FROM order_list a
INNER JOIN
(
SELECT ORDER_DATE, PRODUCT_ID, BUY_NUM FROM
(SELECT DATE_ADD(ORDER_DATE, INTERVAL 1 DAY) ORDER_DATE, PRODUCT_ID, BUY_NUM FROM order_list)x
)b
ON a.ORDER_DATE=b.ORDER_DATE AND a.BUY_NUM > b.BUY_NUM;
```

Figure 42: Selling Amount Analysis