**Faculty of Information**

**INF1343H - Data Modelling and Database Design**

**Assignment 2 Database Design**

**Instructor: Maher Elshakankiri**

**Group 33**
**Student:**
**Yiwen Ma 1009692576**
**Yuetong Jiang 1004339343**
**Keying Zhu 1003790953**

**Table of Content**

**Executive Summary** **3**
**Context for the Study** **3**
**ERD Diagram** **5**
**Tables (ER-to-Relational Mapping)** **6**
**Normalization** **9**
Entity Relations with Composite Attributes 9
Supplier 9
Company-Shein 10
Payment 10
Entity Relations with Multivalued Attributes 11
Employee_Permission 11
Entity Relations in Highest Normal Forms 12
Relationship Relations 12
**Tables (instance)** **12**
**Views (SQL statements and instance)** **20**
**Queries(SQL statements and instance)** **24**
**Triggers (SQL statements and instance)** **29**
Trigger 1: prevent_CEO_deletion 29
Trigger 2: purchase_order_quantity_check 29
Trigger 3: Unit_Price_Check 29
**Database Catalog** **30**
**Discussion on Legal, Ethical & Security Aspects** **32**
**Legal Aspects** **32**
**Ethical Aspects** **35**
**Security Aspects** **37**
**References** **40**
**Statement of Individual Contributions** **41**

## Executive Summary

This report aims to provide a comprehensive practice to apply data modeling and database design techniques to selected organization settings – SHEIN's purchase ordering structure of its Supplier Management System (SRM). The SRM system of SHEIN is an essential component of the company's strategy for optimizing supply chain efficiency and the purchase ordering function plays a critical role in improving productivity and offering end to end purchase order (PO) management. To enable efficient and effective administration of purchase orders, the following data modeling and database design techniques will be presented in this report:

- implement SHEIN's purchasing order database under its SRM system
- make the purchase order database normalized and secured
- convert the ERD diagram into tables, create the tables in MySQL
- implement views, queries, and triggers
- provide a data catalog

The ERD diagram will be revised and updated from assignment 1, then converted into a database schema using the ER-to-Relational mapping tool. The database will be normalized to reduce redundancy and dependency which will help to improve the overall performance, consistency, and accuracy of the database. There are updates and changes that were made due to the process of normalization and a detailed explanation will be provided in the related section. In addition, ten views will be implemented to simplify user data access by hiding the complexity of the underlying data structures. Ten queries will be performed to retrieve data from the database and present it in a format that can be easily understood. Three triggers for stored structure type will be set to the table that is executed automatically in response to specific changes to data in the database. Moreover, a data catalog will be provided to understand the overall information on the database which includes attribute name, type, domain, constraints, and a column description. Finally, a discussion will be presented on legal, ethical, and security aspects related to database management and data usage. With the increasing amount of data being collected and stored, there is a greater need for organizations to ensure that they are adhering to these practices.

## Context for the Study

### About Organization – SHEIN

SHEIN is a rapidly expanding online fashion and lifestyle retailer that has gained tremendous popularity among young people all over the world. It was founded in 2012 by Chris Xu in Nanjing, China and now it's headquartered in Singapore. SHEIN aims to provide affordable yet stylish clothes to all with a mission to bring fashion accessibility to everyone in today's world (SHEIN, 2022). SHEIN believes that everyone, not only the privileged few, should be able to enjoy the beauty of fashion. Nowadays, consumers no longer adhere to a single criterion of what is "fashionable" or "beautiful". SHEIN believes that what you wear represents your personality and encourages everyone to accept and show their individuality (SHEIN, 2022).

The massive fashion retailer has had a tremendous influence on the whole fashion business by offering attractive and fashionable apparel ranging from casual wear to formal wear and catering to all fashion tastes, making it a popular choice among fashion enthusiasts at a reasonable price. The brand has a significant presence on social media channels like Instagram, where they present

their current designs and actively communicate with their followers. Consumers are at the core of SHEIN's business model. As of date, SHEIN has served customers in over 150 countries through its overseas operations, employing approximately 10,000 people and 58% of whom are women (SHEIN, 2022).

## About SRM System

A Supplier Relationship Management (SRM) system is a collection of procedures and technologies that businesses use to successfully manage their interactions with suppliers. It is a comprehensive approach that covers the entire supplier relationship lifecycle, from supplier selection, purchase ordering, performance monitoring to supplier development. The main purpose of using the SRM system is to establish strong and long-term relationships with suppliers to ensure the organization's success in meeting its business objectives. For the vendor side, the system assists firms in reducing costs, improving quality, lowering risks, and increasing innovation and competitiveness. The SRM system helps to connect the organization's aims and goals with those of its suppliers by combining diverse departments and functions, resulting in mutual advantages and value (Groover, 2021). The SRM system has been commonly used across different industries, specifically in the fashion industry where vendors play an important role in a company's business model (Groover, 2021).

## Comparable Organizations in the Fashion Industry

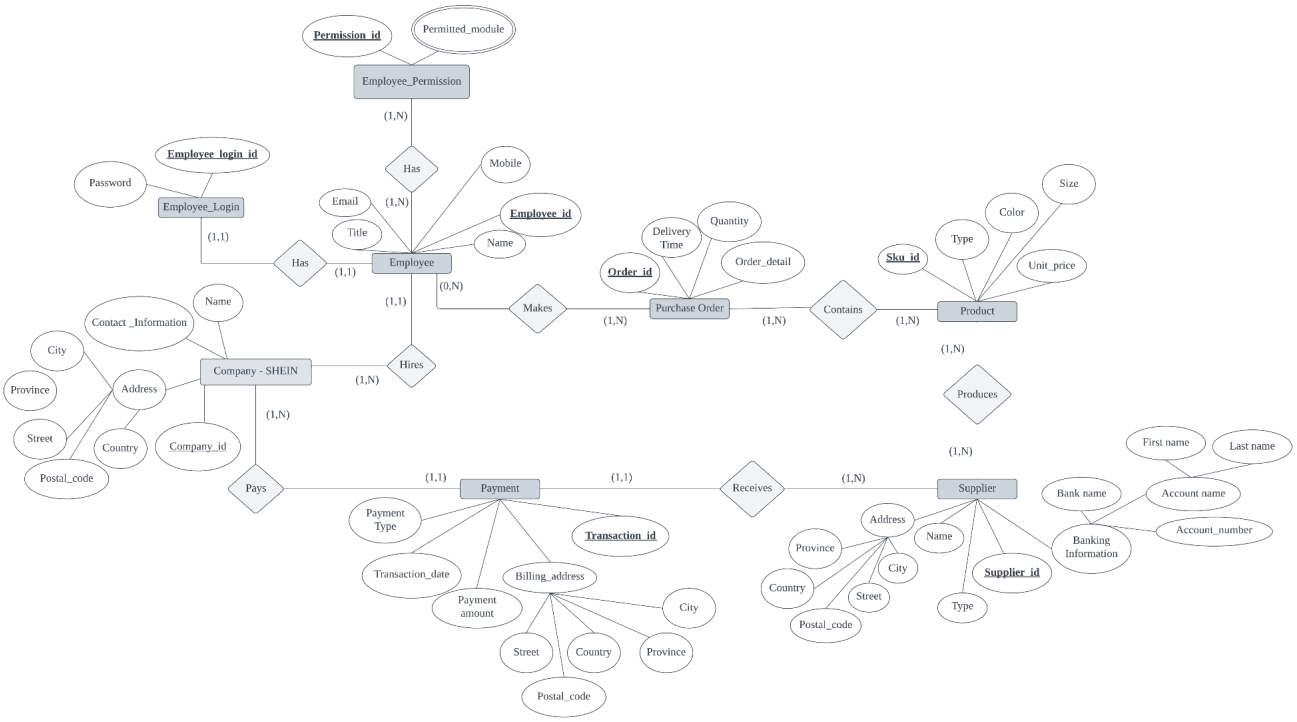
The SRM system is commonly used in the fashion industry where it serves as a communication platform for producers, distributors, and retailers along the whole supply chain. It provides total control over all sourcing and vendor management which includes acquiring initial sourcing intelligence, collecting vendor quotations, monitoring production planning, tracking purchase order progress, and customer compliance audits (DESL, 2023). Zara, one of SHEIN's major competitors has a long-standing reputation for its efficient logistics and the SRM system plays a significant role in achieving this. By leveraging data collected on purchase orders, the SRM system can forecast the precise demand of each product for each shop. Therefore, Zara can be able to send out a small number of products twice a week (Patel, 2023). As a result, it generates a sense of scarcity, with very few unsold things, and if the experiment fails, there is plenty of time for their extremely responsive Supply Chain to test alternative styles. This finally aids Zara in locating the correct goods practically every time (Patel, 2023). Nevertheless, the SRM system is typically purchased from a third party as a basic framework and then modified by users based on their business requirements, which is especially true for large organizations. Zara has constantly evolved its SRM system and numerous self-developed forecast models within the system to optimize their supply efficiency.

## Current SRM System in SHEIN

As a fast-growing fashion retailer giant, SHEIN built its own SRM system based on the third-party framework in 2019 with an in-house product research and development team. This system is linked to the corporate database through the cloud, as well as a physical server within the organization. Employees can access this system via the workplace intranet by entering their

employee IDs. Access to this system has been customized and limited for each employee based on their business team. Employees with permission to make purchase orders can place the order on the system with reference to forecast quantities. Suppliers will receive the order information in real time and be able to prepare and ship the products to the company warehouse. Same as ZARA, SHEIN also has its own demand forecasting model which evolved over time by keeping learning from thousands of purchase orders.

## ERD Diagram




The above ERD illustrates the relationships between entities of the purchasing order in the SRM system. Each entity is explained and described by their own attributes. There are a total of eight entities in this ERD including Company-SHEIN, Payment, Supplier, Product, Purchase order, Employee, Employee_login and Employee_permission. To express the relationship between entities, rectangles are used and verbs are connected with squares and lines. The (min, max) notation for relationship structural constraints also indicated along the line. For instance, between supplier and product entity (1,N) is represented as structural constraints for both sides of the relationship, which means there is at least one product produced by one supplier and a product can be produced by multiple suppliers and vice versa. In addition, the multi-valued attribute is placed in double-lined circles, for instance, the attribute Permission_module of the Employee_Permission entity contains multiple modules (read,write,execute...). Overall, this ERD provides a clear and concise way to understand the relationships and structure of purchasing order data within the SRM system and its database.

## Tables (ER-to-Relational Mapping)

To convert the ERD diagram into MySQL Workbench, we followed the ER-to-Relational Mapping Algorithm. This algorithm has seven steps in total, and in this section, we will show the mapping process step by step. The instance example will be provided after normalization with the table value inserted. See detailed explanation in Table instance section.

**Step 1: Mapping of Regular Entity Types**
The regular entity types in our ERD diagram include Company, Employee, Address, Purchase Order, Product, and Supplier. For each of these entity types, we created a table in our MySQL database with the corresponding attributes and primary keys.

**Step 2: Mapping of Weak Entity Types**
Step 2 of the ER-to-Relational Mapping Algorithm involves mapping weak entity types, but we do not have any weak entities in our ERD diagram. Therefore, we proceeded directly to step 3, which involves mapping associative entity types.

**Step 3: Mapping of Binary 1:1 Relation Types**
In our ERD diagram, we have a one-to-one relationship between Employee and Employee_Login. To map this relationship, we added the foreign key Em_Employee_id to the Employee_login table from the Employee table.

**Step 4: Mapping of Binary 1:N Relationship Types**
We have three one-to-many relationships in our ERD diagram. The first relationship is between Company and Employee. To map this relationship, we added the foreign key C_id in the many-side Employee table, which references the primary keys in the one-side Company-SHEIN table.

The second relationship is between Company and Payment. To map this relationship, we added the foreign key C_Company_id in the many-side Payment table, which references the primary keys in the one-side Company-SHEIN table.

The third relationship is between Supplier and Payment. To map this relationship, we added the foreign key P_Transaction_id in the many-side Supplier table, which references the primary keys in the one-side Payment table.

**Step 5: Mapping of Binary M:N Relationship Types**
We have four many-to-many relationships in our ERD diagram:
The first one is Employee and Purchase_Order. To map this relationship, we created a table Employee_makes_Purchase_Order with foreign keys that reference the primary keys from Employee and Purchase_Order which are E_Employee_id and P_Order_Order_id.

The second one is Purchase_Order and Product. To map this relationship, we created a table Purchase_Order_Contains_Product with foreign keys that reference the primary keys from Purchase_Order and Product which are P_Order_id and P_Sku_id.

The third one is Supplier and Product. To map this relationship, we created a table Supplier_has_Product with foreign keys that reference the primary keys from Supplier and Product which are S_Supplier_id and Product_Sku_id.

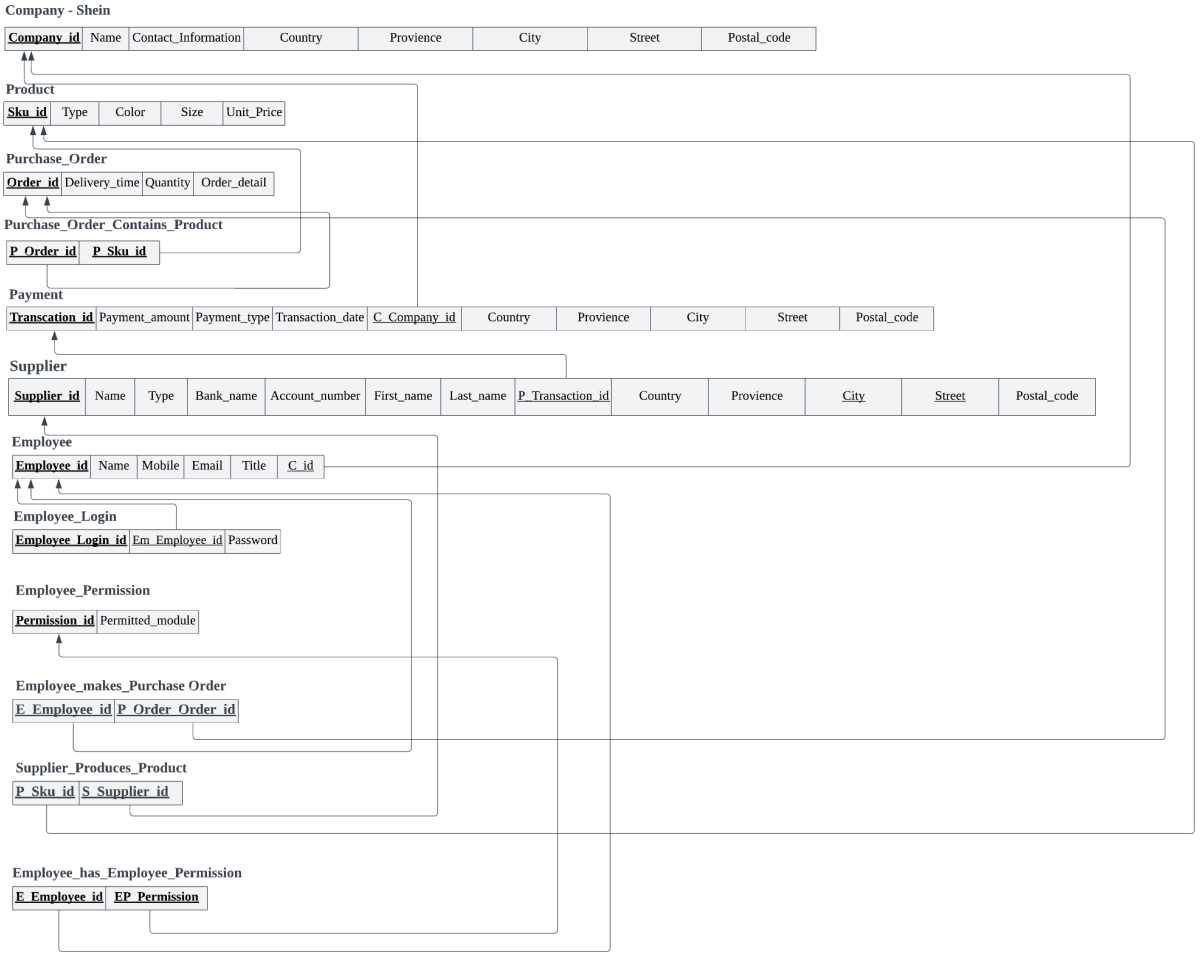
The fourth one is Employee and Employee_Permission. To map this relationship,we created a table Employee_has_Employee_Permission with foreign keys that reference the primary keys from Employee and Employee_Permission which are E_Employee_id and EP_Permission.

**Step 6: Mapping of Multivalued attributes**
According to the ER-to-Relational Mapping Algorithm, each multivalued attribute in a table should be mapped to a new relation with an attribute corresponding to that multivalued attribute. However, in our case, we only have one multivalued attribute Permitted_module in the Employee_Permission table. Since this table already has a primary key attribute "Permission_id" and a foreign key attribute E_Employee_id, we do not need to create a new relation for the multivalued attribute. Therefore, no further mapping is required for the Employee_Permission table.

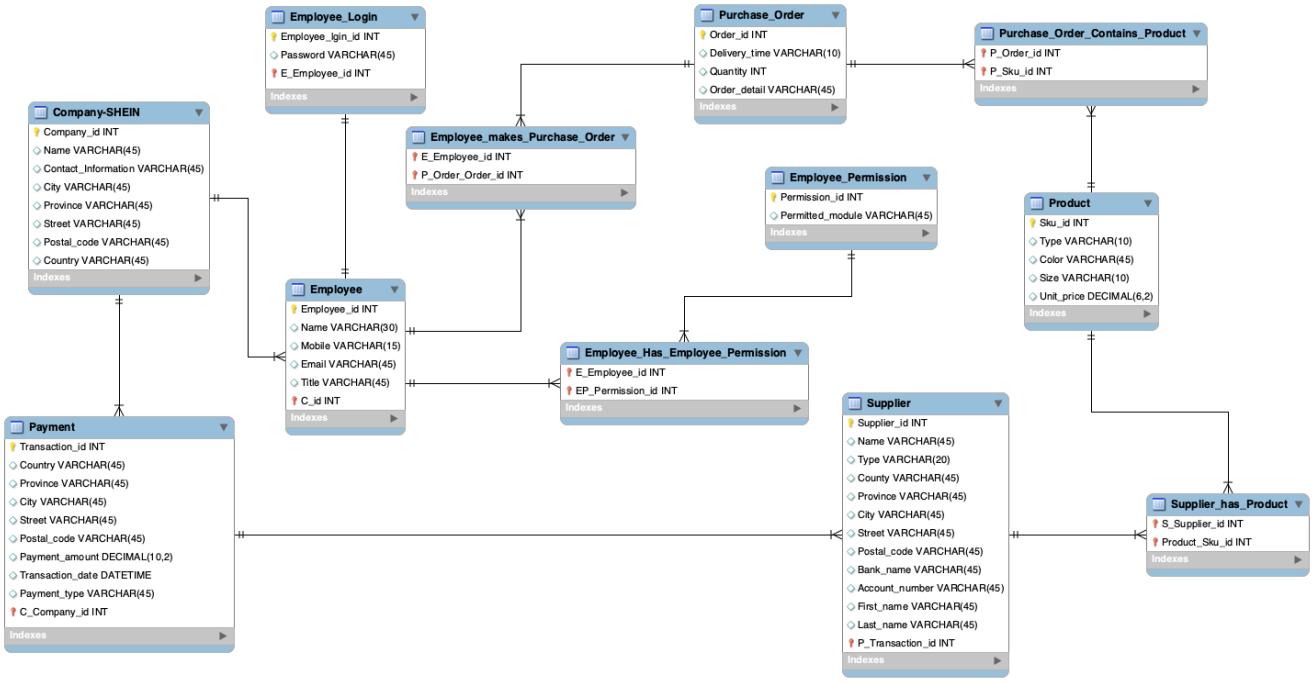
**Step 7: Mapping of N-ary Relationship Types**
Since the model does not include any N-ary relationships, the relationships between entity types are all binary relationships, meaning that they involve two entity types connected by a relationship. Thus, we can apply the ER-to-Relational Mapping Algorithm directly to each binary relationship, without the need for N-ary relationship mapping.

The ER schema is successfully mapped into a relational database schema using the ER-to-Relational Mapping Algorithm. This mapping process resulted in the creation of multiple tables, each with appropriate attributes, primary keys, and foreign keys. Below is the ER schema into a relational database schema.

**Relational Database schema (result of mapping)**




## Database Schema (MySQL)




Since normalization will modify this schema, we will show the updated schema and create the database after normalization.

## Normalization

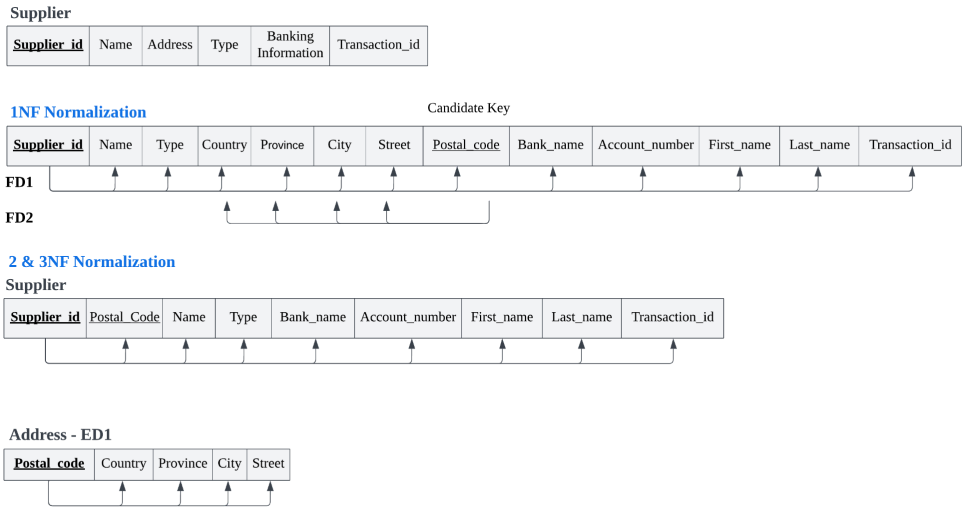
Normalization is a process to reduce redundancy and update anomalies, and improve the quality of database design by formally measuring why one group of attributes in a relational schema could perform better than another. To achieve this, we use constraints and functional dependencies to move from the first normal form (1NF) to the Boyce-Codd normal form (BCNF), which significantly helps to ensure database consistency and save storage space. The normalization process was carried out table by table, until each table reached the highest normal form (BCNF).

## Entity Relations with Composite Attributes

## Supplier

The entity relation supplier has one primary key ‘Supplier_id’ and five non-key attributes, where ‘Address’ and ‘Banking Information’ are composite attributes, while the others are simple attributes. By first NF, we separated the composite attributes into multiple simple attributes. For example, ‘Address’ is split into ‘Country’, ‘Province’, ‘City’, ‘Street’, and ‘Postal_code’, while ‘Banking_information’ is split into ‘Bank_name’, ‘Account_number’, ‘First_name’, and ‘Last_name’. The new attribute ‘Postal_code’ becomes a candidate (secondary key), because the specific address is functionally dependent on the ‘Postal_code’, while ‘Postal_code’ is dependent on the primary key ‘Supplier_id’.

Since we only have only one primary key and no non-prime attribute which depends on the part of the primary keys, the table is also in second normal form 2NF. According to 3NF, there should not be any transitive dependencies in the table. However, we found that ‘Street’, ‘City’, ‘Province’ and ‘Country’ depend on ‘Postal_code’, and ‘Postal_code’ depends on ‘Supplier_id’. As shown in Figure x, we removed the non-key columns which are functionally dependent on ‘Postal_code’ into a new table called ‘Address’.

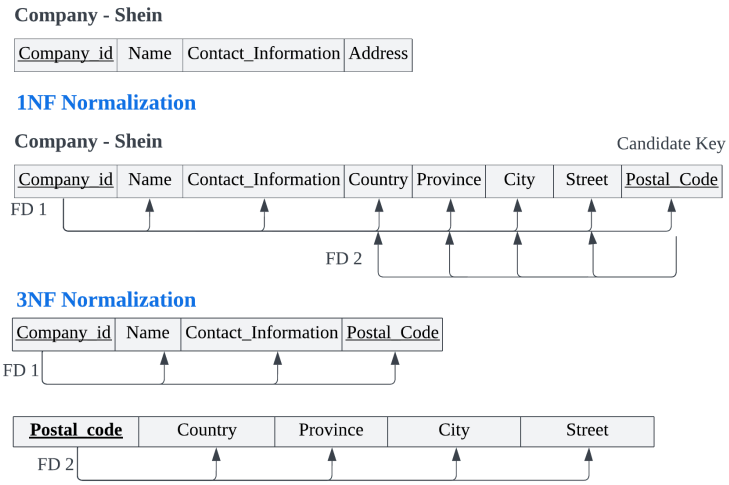



Now, every non-key attribute is fully dependent on the primary key and transitive dependencies are eliminated. At the same time, each column that determines the value of another column is a

candidate key, and all non-key attributes are functionally dependent on the entire set of candidate keys. Therefore, both 3NF and BCNF are done.

## Company-Shein

The entity relation ‘Company-Shein’, where we have one primary key ‘Company_id’, one composite attribute ‘Address’, and two simple attributes. Similarly, to bring the table into 1NF, we separated ‘Address’ into ‘Country’, ‘Province’, ‘City’, ‘Street’, ‘Postal_code’. As there are no partial dependencies, and all non-key attributes are dependent on the entire primary key, 2NF is achieved as well.




Since all the new attributes ‘Street’, ‘City’, ‘Province’ and ‘Country’ are depending on the key ‘Postal_code’, and ‘Postal_code’ is depending on the primary key ‘Company_id’. To achieve 3NF, we eliminated the transitive dependencies by moving these attributes into the new table ‘Address’, as shown below.

## Payment

In the entity relation ‘Payment’, we have a composite attribute ‘Billing Address’. To normalize this relation, we follow the same steps we used for ‘Company’ and ‘Supplier’: separating composite attributes into simple attributes, eliminating partial dependencies and transitive dependencies, and ensuring the determinant is a superkey of the relation. As a result, 1NF, 2NF, 3NF, and BCNF all achieved.

**Payment**

| Transcation_id | Payment_amount | Payment_type | Transaction_date | Billing_Address |
|---|---|---|---|---|

**1NF Normalization**

Candidate Key

| Transcation_id | Payment_amount | Payment_type | Transaction_date | Country | Province | City | Street | Postal_code |
|---|---|---|---|---|---|---|---|---|

FD 1

FD 2

**3NF Normalization**

| Transcation_id | Payment_amount | Payment_type | Transaction_date | Postal_code |
|---|---|---|---|---|

FD 1

| Postal_code | Country | Province | City | Street |
|---|---|---|---|---|

FD 2

The new tables ‘Address’ that we created for ‘Company’, ‘Supplier’, and ‘Payment’ share the same structure, so we only need to include one ‘Address’ table into the relational database. The three tables can then use the foreign key 'Postal_code' to refer to the 'Address' table, which reduces redundancy and saves storage space, while ensuring consistency across the database.

## Entity Relations with Multivalued Attributes

## Employee_Permission

In the entity relation ‘Employee_Permission’, we have one primary key ‘Permision_id’ and one multivalued attribute ‘Permitted_module’. To bring the table into 1NF, we have to split the instances which have multiple values of ‘Permitted_module’ into several rows where each row has only one single value. To ensure the instance with the same primary key has the unique values for all other attributes, we added a new foreign key ‘Employee_id’ which refers to the ‘Employee’ table. In that case, the values in attribute ‘Permision_id’ will remain unique since it’s still the primary key of the ‘Employee_Permission’ table. However, the values in the ‘Employee_id’ may be duplicated, depending on how many permissions each employee has. This modification will bring the table into 1NF, as each attribute has atomic values. Moreover, the table is also in BCNF because there are no partial dependencies (only one primary key attribute) or transitive dependencies, and all functional dependencies are based on candidate keys.

**Employee_Permission**

| Employee_id | Permitted_module |
|---|---|

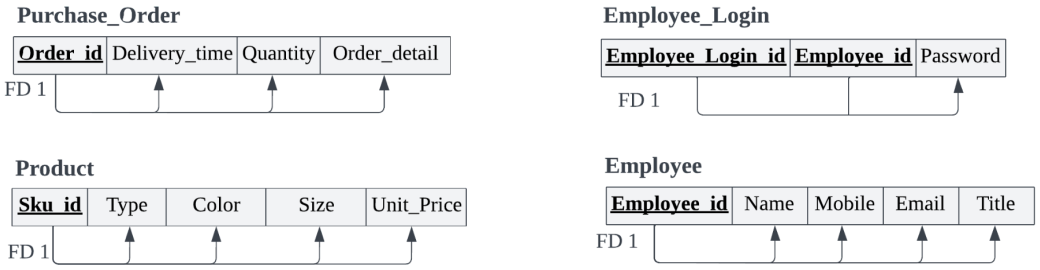
**1NF Normalization**

| Permission_id | Permitted_module |
|---|---|
| 1 | Execute, Read, Write |
| 2 | Write |
| 3 | Read |
| 4 | Read, Write |
| 5 | Execute |

| Permission_id | Employee_id | Permitted_module |
|---|---|---|
| 1 | 1 | Execute |
| 2 | 1 | Read |
| 3 | 1 | Write |
| 4 | 2 | Write |
| 5 | 3 | Read |
| 6 | 4 | Read |
| 7 | 4 | Write |
| 8 | 5 | Execute |

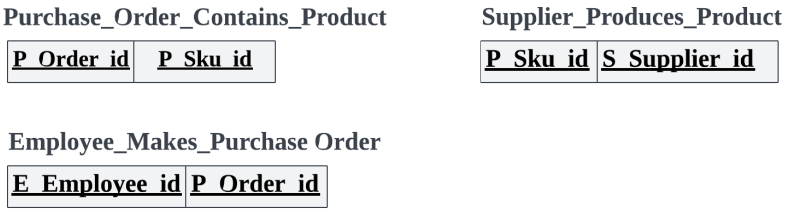
## Entity Relations in Highest Normal Forms

Entity relations ‘Product’, ‘Purchase_Order’, ‘Employee’, and ‘Employee_Login’ have already reached the highest normal form (BCNF), since there is only one key (primary key) in each table and the rests are all simple non-prime attributes. There are no partial dependencies or transitive dependencies, and all functional dependencies are based on the candidate key.



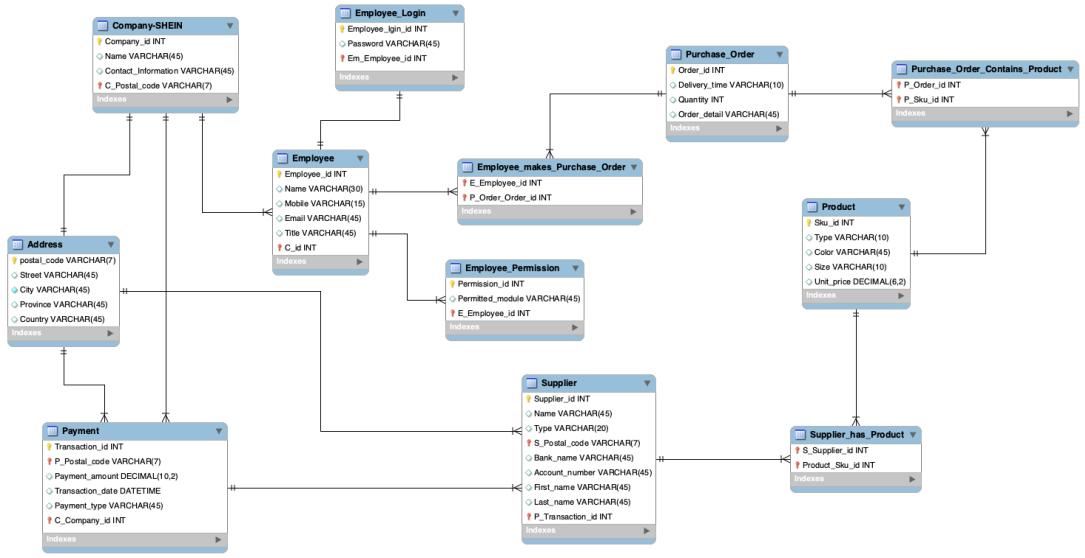

## Relationship Relations

When we conducted the process of ER-to-Relational Mapping, we created three relationship relations: Purchase_Order_Contains_Product, Employee_Makes_Purchase_Order, and Supplier_Produces_Product. Each of these relations contains only two key attributes which are foreign keys referring to the two sides of the relationship they represent. Hence, we do not need to normalize those three relations, as they have already reached the highest normal form.




## Tables (instance)

After fishing the normalization, our new database schema shows below:




And the SQL code below creates several tables for the SHEIN database schema.

```
7
8 •  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
9 •  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
10 •  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
11
12 •  CREATE SCHEMA IF NOT EXISTS `SHEIN` DEFAULT CHARACTER SET utf8 ;
13
14 •  CREATE TABLE IF NOT EXISTS `SHEIN`.`Company-SHEIN` (
15      `Company_id` INT(11) NOT NULL,
16      `Name` VARCHAR(45) NULL DEFAULT NULL,
17      `Contact_Information` VARCHAR(45) NULL DEFAULT NULL,
18      `C_Postal_code` VARCHAR(7) NOT NULL,
19      PRIMARY KEY (`Company_id`, `C_Postal_code`),
20      INDEX `C_Postal_code_idx` (`C_Postal_code` ASC) VISIBLE,
21      CONSTRAINT `C_Postal_code`
22        FOREIGN KEY (`C_Postal_code`)
23        REFERENCES `SHEIN`.`Address` (`postal_code`)
24        ON DELETE CASCADE
25        ON UPDATE CASCADE)
26    ENGINE = InnoDB
27    DEFAULT CHARACTER SET = utf8;
28
29 •  CREATE TABLE IF NOT EXISTS `SHEIN`.`Employee_Login` (
30      `Employee_lgin_id` INT(11) NOT NULL,
31      `Password` VARCHAR(45) NULL DEFAULT NULL,
32      `E_Employee_id` INT(11) NOT NULL,
33      PRIMARY KEY (`Employee_lgin_id`, `E_Employee_id`),
34      INDEX `E_Employee_id_idx` (`E_Employee_id` ASC) VISIBLE,
35      CONSTRAINT `E_Employee_id`
36        FOREIGN KEY (`E_Employee_id`)
37        REFERENCES `SHEIN`.`Employee` (`Employee_id`)
38        ON DELETE CASCADE
39        ON UPDATE CASCADE)
40    ENGINE = InnoDB
41    DEFAULT CHARACTER SET = utf8;
42
```

```
42
43 •  CREATE TABLE IF NOT EXISTS `SHEIN`.`Employee_Permission` (
44      `Permission_id` INT NOT NULL,
45      `Permitted_module` VARCHAR(45) NULL,
46      `E_Employee_id` INT NOT NULL,
47      PRIMARY KEY (`Permission_id`, `E_Employee_id`),
48      INDEX `E_Employee_id_idx` (`E_Employee_id` ASC) VISIBLE,
49      CONSTRAINT `E_Employee_id`
50        FOREIGN KEY (`E_Employee_id`)
51        REFERENCES `SHEIN`.`Employee` (`Employee_id`)
52        ON DELETE CASCADE
53        ON UPDATE CASCADE)
54    ENGINE = InnoDB;
55
56 •  CREATE TABLE IF NOT EXISTS `SHEIN`.`Employee` (
57      `Employee_id` INT(11) NOT NULL,
58      `Name` VARCHAR(30) NULL DEFAULT NULL,
59      `Mobile` VARCHAR(15) NULL DEFAULT NULL,
60      `Email` VARCHAR(45) NULL DEFAULT NULL,
61      `Title` VARCHAR(45) NULL DEFAULT NULL,
62      `C_id` INT(11) NOT NULL,
63      PRIMARY KEY (`Employee_id`, `C_id`),
64      INDEX `C_id _idx` (`C_id` ASC) VISIBLE,
65      CONSTRAINT `C_id `
66        FOREIGN KEY (`C_id`)
67        REFERENCES `SHEIN`.`Company-SHEIN` (`Company_id`)
68        ON DELETE CASCADE
69        ON UPDATE CASCADE)
70    ENGINE = InnoDB
71    DEFAULT CHARACTER SET = utf8;
```

```
72
73  CREATE TABLE IF NOT EXISTS `SHEIN`.`Payment` (
74    `Transaction_id` INT(11) NOT NULL,
75    `P_Postal_code` VARCHAR(7) NOT NULL,
76    `Payment_amount` DECIMAL(10,2) NULL DEFAULT NULL,
77    `Transaction_date` DATETIME NULL DEFAULT NULL,
78    `Payment_type` VARCHAR(45) NULL DEFAULT NULL,
79    `C_Company_id` INT(11) NOT NULL,
80    PRIMARY KEY (`Transaction_id`, `C_Company_id`, `P_Postal_code`),
81    INDEX `C_Company_id_idx` (`C_Company_id` ASC) VISIBLE,
82    INDEX `P_Postal_code_idx` (`P_Postal_code` ASC) VISIBLE,
83    CONSTRAINT `C_Company_id`
84      FOREIGN KEY (`C_Company_id`)
85      REFERENCES `SHEIN`.`Company-SHEIN` (`Company_id`)
86      ON DELETE CASCADE
87      ON UPDATE CASCADE,
88    CONSTRAINT `P_Postal_code`
89      FOREIGN KEY (`P_Postal_code`)
90      REFERENCES `SHEIN`.`Address` (`postal_code`)
91      ON DELETE NO ACTION
92      ON UPDATE NO ACTION)
93  ENGINE = InnoDB
94  DEFAULT CHARACTER SET = utf8;
95
96  CREATE TABLE IF NOT EXISTS `SHEIN`.`Purchase_Order` (
97    `Order_id` INT(11) NOT NULL,
98    `Delivery_time` VARCHAR(10) NULL DEFAULT NULL,
99    `Quantity` INT(11) NULL DEFAULT NULL,
100   `Order_detail` VARCHAR(45) NULL DEFAULT NULL,
101   PRIMARY KEY (`Order_id`))
102 ENGINE = InnoDB
103 DEFAULT CHARACTER SET = utf8;
104
105 CREATE TABLE IF NOT EXISTS `SHEIN`.`Product` (
106   `Sku_id` INT(11) NOT NULL,
107   `Type` VARCHAR(10) NULL DEFAULT NULL,
108   `Color` VARCHAR(45) NULL DEFAULT NULL,
109   `Size` VARCHAR(10) NULL DEFAULT NULL,
110   `Unit_price` DECIMAL(6,2) NULL DEFAULT NULL,
111   PRIMARY KEY (`Sku_id`))
112 ENGINE = InnoDB
113 DEFAULT CHARACTER SET = utf8;
114
115 CREATE TABLE IF NOT EXISTS `SHEIN`.`Supplier` (
116   `Supplier_id` INT(11) NOT NULL,
117   `Name` VARCHAR(45) NULL DEFAULT NULL,
118   `Type` VARCHAR(20) NULL DEFAULT NULL,
119   `S_Postal_code` VARCHAR(7) NOT NULL,
120   `Bank_name` VARCHAR(45) NULL DEFAULT NULL,
121   `Account_number` VARCHAR(45) NULL DEFAULT NULL,
122   `First_name` VARCHAR(45) NULL DEFAULT NULL,
123   `Last_name` VARCHAR(45) NULL DEFAULT NULL,
124   `P_Transaction_id` INT(11) NOT NULL,
125   PRIMARY KEY (`Supplier_id`, `P_Transaction_id`, `S_Postal_code`),
126   INDEX `P_Transaction_id_idx` (`P_Transaction_id` ASC) VISIBLE,
127   INDEX `S_Postal_code_idx` (`S_Postal_code` ASC) VISIBLE,
128   CONSTRAINT `P_Transaction_id`
129     FOREIGN KEY (`P_Transaction_id`)
130     REFERENCES `SHEIN`.`Payment` (`Transaction_id`)
131     ON DELETE CASCADE
132     ON UPDATE CASCADE,
133   CONSTRAINT `S_Postal_code`
134     FOREIGN KEY (`S_Postal_code`)
135     REFERENCES `SHEIN`.`Address` (`postal_code`)
136     ON DELETE NO ACTION
137     ON UPDATE CASCADE)
138 ENGINE = InnoDB
139 DEFAULT CHARACTER SET = utf8;
```

```
140
141 • CREATE TABLE IF NOT EXISTS `SHEIN`.`Employee_makes_Purchase_Order` (
142     `E_Employee_id` INT(11) NOT NULL,
143     `P_Order_Order_id` INT(11) NOT NULL,
144     PRIMARY KEY (`E_Employee_id`, `P_Order_Order_id`),
145     INDEX `fk_Employee_has_Purchase_Order_Purchase_Order1_idx` (`P_Order_Order_id` ASC) VISIBLE,
146     INDEX `fk_Employee_has_Purchase_Order_Employee1_idx` (`E_Employee_id` ASC) VISIBLE,
147     CONSTRAINT `fk_Employee_has_Purchase_Order_Employee1`
148       FOREIGN KEY (`E_Employee_id`)
149       REFERENCES `SHEIN`.`Employee` (`Employee_id`)
150       ON DELETE CASCADE
151       ON UPDATE CASCADE,
152     CONSTRAINT `fk_Employee_has_Purchase_Order_Purchase_Order1`
153       FOREIGN KEY (`P_Order_Order_id`)
154       REFERENCES `SHEIN`.`Purchase_Order` (`Order_id`)
155       ON DELETE CASCADE
156       ON UPDATE CASCADE)
157   ENGINE = InnoDB
158   DEFAULT CHARACTER SET = utf8;
159
160 • CREATE TABLE IF NOT EXISTS `SHEIN`.`Purchase_Order_Contains_Product` (
161     `P_Order_id` INT(11) NOT NULL,
162     `P_Sku_id` INT(11) NOT NULL,
163     PRIMARY KEY (`P_Order_id`, `P_Sku_id`),
164     INDEX `fk_Purchase_Order_has_Product_Product1_idx` (`P_Sku_id` ASC) VISIBLE,
165     INDEX `fk_Purchase_Order_has_Product_Purchase_Order1_idx` (`P_Order_id` ASC) VISIBLE,
166     CONSTRAINT `fk_Purchase_Order_has_Product_Purchase_Order1`
167       FOREIGN KEY (`P_Order_id`)
168       REFERENCES `SHEIN`.`Purchase_Order` (`Order_id`)
169       ON DELETE CASCADE
170       ON UPDATE CASCADE,
171     CONSTRAINT `fk_Purchase_Order_has_Product_Product1`
172       FOREIGN KEY (`P_Sku_id`)
173       REFERENCES `SHEIN`.`Product` (`Sku_id`)
174       ON DELETE CASCADE
175       ON UPDATE CASCADE)

176   ENGINE = InnoDB
177   DEFAULT CHARACTER SET = utf8;
178
179 • CREATE TABLE IF NOT EXISTS `SHEIN`.`Supplier_has_Product` (
180     `S_Supplier_id` INT(11) NOT NULL,
181     `Product_Sku_id` INT(11) NOT NULL,
182     PRIMARY KEY (`S_Supplier_id`, `Product_Sku_id`),
183     INDEX `fk_Supplier_has_Product_Product1_idx` (`Product_Sku_id` ASC) VISIBLE,
184     INDEX `fk_Supplier_has_Product_Supplier1_idx` (`S_Supplier_id` ASC) VISIBLE,
185     CONSTRAINT `fk_Supplier_has_Product_Supplier1`
186       FOREIGN KEY (`S_Supplier_id`)
187       REFERENCES `SHEIN`.`Supplier` (`Supplier_id`)
188       ON DELETE CASCADE
189       ON UPDATE CASCADE,
190     CONSTRAINT `fk_Supplier_has_Product_Product1`
191       FOREIGN KEY (`Product_Sku_id`)
192       REFERENCES `SHEIN`.`Product` (`Sku_id`)
193       ON DELETE CASCADE
194       ON UPDATE CASCADE)
195   ENGINE = InnoDB
196   DEFAULT CHARACTER SET = utf8;
197
198 • CREATE TABLE IF NOT EXISTS `SHEIN`.`Address` (
199     `postal_code` VARCHAR(7) NOT NULL,
200     `Street` VARCHAR(45) NULL DEFAULT NULL,
201     `City` VARCHAR(45) NOT NULL,
202     `Province` VARCHAR(45) NULL DEFAULT NULL,
203     `Country` VARCHAR(45) NULL DEFAULT NULL,
204     PRIMARY KEY (`postal_code`))
205   ENGINE = InnoDB
206   DEFAULT CHARACTER SET = utf8;
```

The SQL code below inserts data into tables for the SHEIN database.

```sql
INSERT INTO Address (postal_code, Street, City, Province, Country)
VALUES ( 'A1A 2B2', '1 One Street', 'Toronto', 'ON', 'Canada'),
('A1A 2B1', '65 Data Street', 'Toronto', 'ON', 'Canada'),
('A2A 2B3', '50 SQL Street', 'Toronto', 'ON', 'Canada'),
('A2A 2B7', '1100 Normal Street', 'Toronto', 'ON', 'Canada'),
('A4A 7B8', '120 Trigger Street', 'Toronto', 'ON', 'Canada'),
('M1M 3A3','1 Yonge Street','Toronto', 'ON', 'Canada'),
('M1S 3A3', '12 Yonge Street', 'Toronto', 'ON', 'Canada'),
('M1T 3L3', '13 Yonge Street', 'Toronto', 'ON', 'Canada'),
('M1E 3B3', '145 Yonge Street', 'Toronto', 'ON', 'Canada'),
('M1S 3A4', '167 Yonge Street', 'Toronto', 'ON', 'Canada');

INSERT INTO `Company-SHEIN` (Company_id, Name, Contact_Information, C_Postal_code)
VALUES (1, 'SHEIN', 'Judy.Ma@shein.com','A1A 2B2');

INSERT INTO Employee (Employee_id, Name, Mobile, Email, Title, C_id)
VALUES (1000, 'Judy Ma', '6471111111', 'Judy.Ma@shein.com', 'VP', 1),
(1001, 'Yiwen Ma', '6472222222', 'Yiwen.Ma@shein.com', 'Manager', 1),
(1002, 'Yuetong Jiang', '6473333333', 'Yuetong.Jiang@shein.com', 'Officer', 1),
(1003, 'Keying Zhu', '6474444444', 'Keying.Zhu@shein.com', 'CEO', 1),
(1004, 'Vvan Zhu', '6475555555', 'Vvan.Zhu@shein.com', 'Software Developer', 1);

INSERT INTO Employee_Permission (Permission_id, Permitted_module,E_Employee_id)
VALUES (1, 'Execute',1),
(2, 'Read',1),
(3, 'Write',1),
(4, 'Write',2),
(5, 'Read',3),
(6, 'Read',4),
(7, 'Write',4),
(8, 'Execute',5);

INSERT INTO Employee_Has_Employee_Permission (E_Employee_id, EP_Permission_id)
VALUES (1000, 1),
(1001, 2),
(1002, 3),
(1003, 4),
(1004, 5);

INSERT INTO Employee_Login (Employee_lgin_id, Password, E_Employee_id)
VALUES (23, 'abc123456', 1000),
(65, 'def654321', 1001),
(17, 'hij246810', 1002),
(13, 'klm13579', 1003),
(3, 'opq09876', 1004);

INSERT INTO Purchase_Order (Order_id, Delivery_time, Quantity, Order_detail)
VALUES (100001, '24h', 190, 'shirts'),
(100002, '24h', 191, 'pants'),
(100003, '48h', 192, 'sneakers'),
(100004, '72h', 193, 'tops'),
(100005, '24h', 194, 'sweaters');

INSERT INTO Product (Sku_id, Type, Color, Size, Unit_price)
VALUES (14623, 'Men', 'blue', 'M', 39.9),
(21769, 'Men', 'black', 'M', 39.9),
(47123, 'Women', 'green', 'S', 39.9),
(10771, 'Kid', 'white', 'S', 29.9),
(21009, 'Accesorios', 'blue', 'NA', 19.9);

INSERT INTO Purchase_Order_Contains_Product (P_Order_id, P_Sku_id)
VALUES (100001, 14623),
(100002, 21769),
(100003, 47123),
(100004, 10771),
(100005, 21009);

INSERT INTO Employee_makes_Purchase_Order (E_Employee_id, P_Order_Order_id)
VALUES (1001, 100001),
(1001, 100002),
(1001, 100003),
(1003, 100004),
```

```
    (1003, 100005);

INSERT INTO Payment (Transaction_id, P_Postal_code, Payment_amount, Transaction_date, Payment_type, C_Company_id)
VALUE (201636, 'A1A 2B2', 12600, '2016-03-06 12:30:00', 'Once', 1),
(201488, 'A1A 2B1', 12600, '2014-08-08 12:30:00', 'Once', 1),
(201948, 'A2A 2B3', 12600, '2019-04-08 12:30:00', 'Once', 1),
(201928, 'A2A 2B7', 12600, '2019-02-08 12:30:00', 'Once', 1),
(202322, 'A4A 7B8', 12600, '2023-02-02 12:30:00', 'Once', 1);

INSERT INTO Supplier (Supplier_id, Name, Type, S_Postal_code, Bank_name, Account_number, First_name, Last_name, P_Transaction_id)
VALUES (110001, 'CompanyA', 'Corporation', 'M1M 3A3', 'CIBC', '711000', 'Ju', 'Mie', 201636),
(110002, 'CompanyB', 'Partnership', 'M1S 3A3', 'RBC', '711000', 'Juliet', 'Mia', 201488),
(110003, 'CompanyC', 'Sole Proprietorship', 'M1T 3L3', 'BMO', '711000', 'Key', 'Zhu', 201948),
(110004, 'CompanyD', 'Corporation', 'M1E 3B3', 'Scotia', '711000', 'Lock', 'Zhu', 201928),
(110005, 'CompanyE', 'Corporation', 'M1S 3A4', 'HSBC', '711000', 'Ivey', 'Western', 202322);

INSERT INTO Supplier_has_Product (S_Supplier_id, Product_Sku_id)
VALUES (110001, 14623),
(110002, 21769),
(110003, 47123),
(110004, 10771),
(110005, 21009);
```

Here is the overview of each table, more detailed information will be showed on the Database Catalog section, and the table, after data insertion, appears as follows:

Company-SHEIN: This table stores company information such as the company's name, contact information, and the related postal code.

Company-SHEIN

| Company_id | Name | Contact_Information | C_Postal_code |
|---|---|---|---|
| 1 | SHEIN | Judy.Ma@shein.com | A1A 2B2 |

Employee_Login: This table stores employee login information, including employee login IDs and passwords.

Employee_Login

| Employee_lgin_id | Password | Em_Employee_id |
|---|---|---|
| 3 | opq09876 | 1004 |
| 13 | klm13579 | 1003 |
| 17 | hij246810 | 1002 |
| 23 | abc123456 | 1000 |
| 65 | def654321 | 1001 |

Employee_Permission: This table contains information about each employee's permissions, such as permitted modules.

Employee_Permission

| Permission_id | Permitted_module | E_Employee_id |
|---|---|---|
| 1 | Execute | 1000 |
| 2 | Read | 1000 |
| 3 | Write | 1000 |
| 4 | Write | 1001 |
| 5 | Read | 1002 |
| 6 | Read | 1003 |
| 7 | Write | 1003 |
| 8 | Execute | 1004 |

Employee: This table stores employee information, including their name, mobile number, email, job title, and the company they work for.

Employee

| Employee_id | Name | Mobile | Email | Title | C_id |
|---|---|---|---|---|---|
| 1000 | Judy Ma | 6471111111 | Judy.Ma@shein.com | VP | 1 |
| 1001 | Yiwen Ma | 6472222222 | Yiwen.Ma@shein.com | Manager | 1 |
| 1002 | Yuetong Jiang | 6473333333 | Yuetong.Jiang@shein.com | Officer | 1 |
| 1003 | Keying Zhu | 6474444444 | Keying.Zhu@shein.com | CEO | 1 |
| 1004 | Vvan Zhu | 6475555555 | Vvan.Zhu@shein.com | Software Developer | 1 |

Payment: This table holds payment transaction information, including the transaction ID, payment amount, transaction date, payment type, and the company involved in the transaction.

Payment

| Transaction_id | P_Postal_code | Payment_amount | Transaction_date | Payment_type | C_Company_id |
|---|---|---|---|---|---|
| 201488 | A1A 2B1 | 12600.00 | 2014-08-08 12:30:00 | Once | 1 |
| 201636 | A1A 2B2 | 12600.00 | 2016-03-06 12:30:00 | Once | 1 |
| 201928 | A2A 2B7 | 12600.00 | 2019-02-08 12:30:00 | Once | 1 |
| 201948 | A2A 2B3 | 12600.00 | 2019-04-08 12:30:00 | Once | 1 |
| 202322 | A4A 7B8 | 12600.00 | 2023-02-02 12:30:00 | Once | 1 |

Purchase_Order: This table stores information about purchase orders, such as order ID, delivery time, quantity, and order details.

Purchase_Order

| Order_id | Delivery_time | Quantity | Order_detail |
|---|---|---|---|
| 100001 | 24h | 190 | shirts |
| 100002 | 24h | 191 | pants |
| 100003 | 48h | 192 | sneakers |
| 100004 | 72h | 193 | tops |
| 100005 | 24h | 194 | sweaters |

Product: This table contains information about products, including the SKU ID, product type, color, size, and unit price.

Product

| Sku_id | Type | Color | Size | Unit_price |
|---|---|---|---|---|
| 10771 | Kid | white | S | 29.90 |
| 14623 | Men | blue | M | 39.90 |
| 21009 | Accesorios | blue | NA | 19.90 |
| 21769 | Men | black | M | 39.90 |
| 47123 | Women | green | S | 39.90 |

Supplier: This table holds supplier information, including their name, type, postal code, bank name, account number, and related transaction IDs.

Supplier

| Supplier_id | Name | Type | S_Postal_code | Bank_name | Account_number | First_name | Last_name | P_Transaction_id |
|---|---|---|---|---|---|---|---|---|
| **110001** | CompanyA | Corporation | M1M 3A3 | CIBC | 711000 | Ju | Mie | 201636 |
| **110002** | CompanyB | Partnership | M1S 3A3 | RBC | 711000 | Juliet | Mia | 201488 |
| **110003** | CompanyC | Sole Proprietorship | M1T 3L3 | BMO | 711000 | Key | Zhu | 201948 |
| **110004** | CompanyD | Corporation | M1E 3B3 | Scotia | 711000 | Lock | Zhu | 201928 |
| **110005** | CompanyE | Corporation | M1S 3A4 | HSBC | 711000 | Ivey | Western | 202322 |

Employee_makes_Purchase_Order: This table associates employees with the purchase orders they created.

Employee_makes_Purchase_Order

| E_Employee_id | P_Order_Order_id |
|---|---|
| **1001** | 100001 |
| **1001** | 100002 |
| **1001** | 100003 |
| **1003** | 100004 |
| **1003** | 100005 |

Purchase_Order_Contains_Product: This table links purchase orders to the products they contain.

Purchase_Order_Contains_Product

| P_Order_id | P_Sku_id |
|---|---|
| **100004** | 10771 |
| **100001** | 14623 |
| **100005** | 21009 |
| **100002** | 21769 |
| **100003** | 47123 |

Supplier_has_Product: This table associates suppliers with the products they supply.

Supplier_has_Product

| S_Supplier_id | Product_Sku_id |
|---|---|
| **110004** | 10771 |
| **110001** | 14623 |
| **110005** | 21009 |
| **110002** | 21769 |
| **110003** | 47123 |

Address: This table stores address information, including postal code, street, city, province, and country.

Address

| postal_code | Street | City | Province | Country |
|---|---|---|---|---|
| **A1A 2B1** | 65 Data Street | Toronto | ON | Canada |
| **A1A 2B2** | 1 One Street | Toronto | ON | Canada |
| **A2A 2B3** | 50 SQL Street | Toronto | ON | Canada |
| **A2A 2B7** | 1100 Normal Street | Toronto | ON | Canada |
| **A4A 7B8** | 120 Trigger Street | Toronto | ON | Canada |
| **M1E 3B3** | 145 Yonge Street | Toronto | ON | Canada |
| **M1M 3A3** | 1 Yonge Street | Toronto | ON | Canada |
| **M1S 3A3** | 12 Yonge Street | Toronto | ON | Canada |
| **M1S 3A4** | 167 Yonge Street | Toronto | ON | Canada |
| **M1T 3L3** | 13 Yonge Street | Toronto | ON | Canada |

## Views (SQL statements and instance)

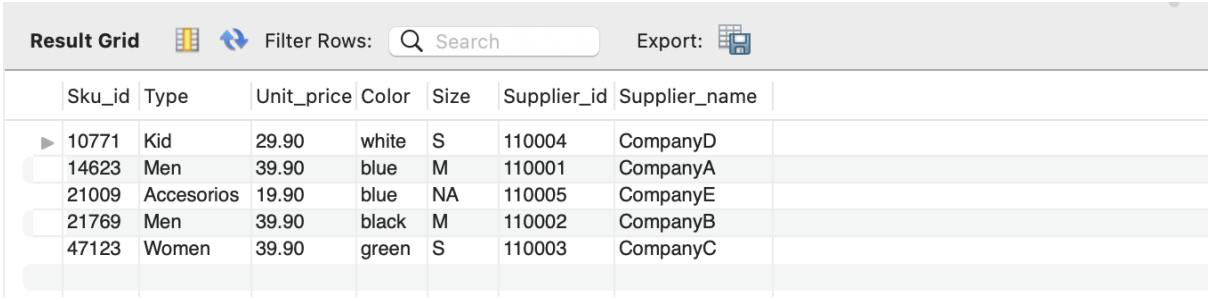
In this SHEIN database, ten views have been created to provide an efficient way to access and display data from multiple tables. These views aggregate, filter, and format the information for specific purposes. We will describe each of them in the following:

**products_with_suppliers:** This view combined product and supplier information, listing each product's SKU, type, unit price, color, size, and corresponding supplier details.

Statement:

```
CREATE VIEW products_with_suppliers AS
SELECT p.Sku_id, p.Type, p.Unit_price, p.Color, p.Size, s.Supplier_id, s.Name AS Supplier_name
FROM Product p
JOIN Supplier_has_Product spp ON p.Sku_id = spp.Product_Sku_id
JOIN Supplier s ON spp.S_Supplier_id = s.Supplier_id;
SELECT * FROM products_with_suppliers;
```

Instance:




| Sku_id | Type | Unit_price | Color | Size | Supplier_id | Supplier_name |
|---|---|---|---|---|---|---|
| 10771 | Kid | 29.90 | white | S | 110004 | CompanyD |
| 14623 | Men | 39.90 | blue | M | 110001 | CompanyA |
| 21009 | Accesorios | 19.90 | blue | NA | 110005 | CompanyE |
| 21769 | Men | 39.90 | black | M | 110002 | CompanyB |
| 47123 | Women | 39.90 | green | S | 110003 | CompanyC |

**employee_purchase_orders:** This view displays employee and purchase order details, including the employee's name and the order's details, quantity, and delivery time.

Statement:

```
CREATE VIEW employee_purchase_orders AS
SELECT e.Employee_id, e.Name AS Employee_name, po.Order_id, po.Order_detail, po.Quantity, po.Delivery_time
FROM Employee e
JOIN Employee_makes_Purchase_Order empo ON e.Employee_id = empo.E_Employee_id
JOIN Purchase_Order po ON empo.P_Order_Order_id = po.Order_id;
SELECT * FROM employee_purchase_orders;
```

Instance:

Result Grid Filter Rows: Search Export:

| Employee_id | Employee_name | Order_id | Order_detail | Quantity | Delivery_time |
|---|---|---|---|---|---|
| 1001 | Yiwen Ma | 100001 | shirts | 190 | 24h |
| 1001 | Yiwen Ma | 100002 | pants | 191 | 24h |
| 1001 | Yiwen Ma | 100003 | sneakers | 192 | 48h |
| 1003 | Keying Zhu | 100004 | tops | 193 | 72h |
| 1003 | Keying Zhu | 100005 | sweaters | 194 | 24h |

**purchase_order_product:** This view lists purchase order information along with associated product data, such as SKU, type, unit price, color, and size.

Statement:

```
CREATE VIEW purchase_order_product AS
SELECT po.Order_id, p.Sku_id, p.Type, p.Unit_price, p.Color, p.Size, po.Quantity
FROM Purchase_Order po
JOIN Purchase_Order_Contains_Product pocp ON po.Order_id = pocp.P_Order_id
JOIN Product p ON pocp.P_Sku_id = p.Sku_id;
SELECT * FROM purchase_order_product;
```

Instance:

Result Grid Filter Rows: Search Export:

| Order_id | Sku_id | Type | Unit_price | Color | Size | Quantity |
|---|---|---|---|---|---|---|
| 100001 | 14623 | Men | 39.90 | blue | M | 190 |
| 100002 | 21769 | Men | 39.90 | black | M | 191 |
| 100003 | 47123 | Women | 39.90 | green | S | 192 |
| 100004 | 10771 | Kid | 29.90 | white | S | 193 |
| 100005 | 21009 | Accesorios | 19.90 | blue | NA | 194 |

**total_products_sold_by_supplier:** This view calculates the total number of products sold by each supplier.

Statement:

```
CREATE VIEW total_products_sold_by_supplier AS
SELECT s.Supplier_id, s.Name AS Supplier_name, COUNT(pocp.P_Sku_id) AS Total_products_sold
FROM Supplier s
JOIN Supplier_has_Product spp ON s.Supplier_id = spp.S_Supplier_id
JOIN Purchase_Order_Contains_Product pocp ON spp.Product_Sku_id = pocp.P_Sku_id
GROUP BY s.Supplier_id, s.Name;
SELECT * FROM total_products_sold_by_supplier;
```

Instance:

Result Grid Filter Rows: Search Export:

| Supplier_id | Supplier_name | Total_products_s... |
|---|---|---|
| 110001 | CompanyA | 1 |
| 110002 | CompanyB | 1 |
| 110003 | CompanyC | 1 |
| 110004 | CompanyD | 1 |
| 110005 | CompanyE | 1 |

**supplier_transactions:** This view shows supplier transaction details, including payment amount, payment type, and transaction date.

Statement:

```
CREATE VIEW supplier_transactions AS
SELECT s.Supplier_id, s.Name AS Supplier_name, p.Transaction_id, p.Payment_amount, p.Payment_type, p.Transaction_date
FROM Supplier s
JOIN Payment p ON s.P_Transaction_id = p.Transaction_id;
SELECT * FROM supplier_transactions;
```

Instance:

Result Grid Filter Rows: Search Export:

| Supplier_id | Supplier_name | Transaction_... | Payment_amount | Payment_type | Transaction_date |
|---|---|---|---|---|---|
| 110001 | CompanyA | 201636 | 12600.00 | Once | 2016-03-06 12:30:00 |
| 110002 | CompanyB | 201488 | 12600.00 | Once | 2014-08-08 12:30:00 |
| 110003 | CompanyC | 201948 | 12600.00 | Once | 2019-04-08 12:30:00 |
| 110004 | CompanyD | 201928 | 12600.00 | Once | 2019-02-08 12:30:00 |
| 110005 | CompanyE | 202322 | 12600.00 | Once | 2023-02-02 12:30:00 |

**employees_with_login:** This view displays employee details along with their corresponding login credentials.

Statement:

```
CREATE VIEW employees_with_login AS
SELECT e.Employee_id, e.Name, e.Email, e.Mobile, e.Title, el.Employee_lgin_id, el.Password
FROM Employee e
JOIN Employee_Login el ON e.Employee_id = el.Em_Employee_id;
SELECT * FROM employees_with_login;
```

Instance:

Result Grid Filter Rows: Search Export:

| Employee_id | Name | Email | Mobile | Title | Employee_lgin_id | Password |
|---|---|---|---|---|---|---|
| 1000 | Judy Ma | Judy.Ma@shein.com | 6471111111 | VP | 23 | abc123456 |
| 1001 | Yiwen Ma | Yiwen.Ma@shein.com | 6472222222 | Manager | 65 | def654321 |
| 1002 | Yuetong Jiang | Yuetong.Jiang@shein.com | 6473333333 | Officer | 17 | hij246810 |
| 1003 | Keying Zhu | Keying.Zhu@shein.com | 6474444444 | CEO | 13 | klm13579 |
| 1004 | Vvan Zhu | Vvan.Zhu@shein.com | 6475555555 | Software Developer | 3 | opq09876 |

**employees_with_login_and_permission:** This view combines employee, login, and permission information, showing each employee's permitted modules.

Statement:

```
CREATE VIEW employees_with_login_and_permission AS
SELECT e.Employee_id, e.Name, e.Email, e.Mobile, e.Title, el.Employee_lgin_id, el.Password, ep.Permitted_module
FROM Employee e
JOIN Employee_Login el ON e.Employee_id = el.Em_Employee_id
JOIN Employee_Permission ep ON e.Employee_id = ep.E_Employee_id;
SELECT * FROM employees_with_login_and_permission;
```

Instance:

Result Grid Filter Rows: Search Export:

| Employee_id | Name | Email | Mobile | Title | Employee_lgin_id | Password | Permitted_mod... |
|---|---|---|---|---|---|---|---|
| 1000 | Judy Ma | Judy.Ma@shein.com | 6471111111 | VP | 23 | abc123456 | Execute |
| 1000 | Judy Ma | Judy.Ma@shein.com | 6471111111 | VP | 23 | abc123456 | Read |
| 1000 | Judy Ma | Judy.Ma@shein.com | 6471111111 | VP | 23 | abc123456 | Write |
| 1001 | Yiwen Ma | Yiwen.Ma@shein.com | 6472222222 | Manager | 65 | def654321 | Write |
| 1002 | Yuetong Jiang | Yuetong.Jiang@shein.com | 6473333333 | Officer | 17 | hij246810 | Read |
| 1003 | Keying Zhu | Keying.Zhu@shein.com | 6474444444 | CEO | 13 | klm13579 | Read |
| 1003 | Keying Zhu | Keying.Zhu@shein.com | 6474444444 | CEO | 13 | klm13579 | Write |
| 1004 | Vvan Zhu | Vvan.Zhu@shein.com | 6475555555 | Software Developer | 3 | opq09876 | Execute |

**purchase_orders_with_products_and_suppliers:** This view combines purchase order, product, and supplier details, listing the order's details, delivery time, and corresponding product and supplier information.

Statement:

```
CREATE VIEW purchase_orders_with_products_and_suppliers AS
SELECT po.Order_id, po.Order_detail, po.Quantity, po.Delivery_time,
       p.Sku_id, p.Type, p.Unit_price, p.Color, p.Size,
       s.Supplier_id, s.Name AS Supplier_name
FROM Purchase_Order po
JOIN Purchase_Order_Contains_Product pocp ON po.Order_id = pocp.P_Order_id
JOIN Product p ON pocp.P_Sku_id = p.Sku_id
JOIN Supplier_has_Product shp ON p.Sku_id = shp.Product_Sku_id
JOIN Supplier s ON shp.S_Supplier_id = s.Supplier_id;
SELECT * FROM purchase_orders_with_products_and_suppliers;
```

Instance:

Result Grid Filter Rows: Search Export:

| Order_id | Order_detail | Quantity | Delivery_time | Sku_id | Type | Unit_price | Color | Size | Supplier_id | Supplier_name |
|---|---|---|---|---|---|---|---|---|---|---|
| 100001 | shirts | 190 | 24h | 14623 | Men | 39.90 | blue | M | 110001 | CompanyA |
| 100002 | pants | 191 | 24h | 21769 | Men | 39.90 | black | M | 110002 | CompanyB |
| 100003 | sneakers | 192 | 48h | 47123 | Women | 39.90 | green | S | 110003 | CompanyC |
| 100004 | tops | 193 | 72h | 10771 | Kid | 29.90 | white | S | 110004 | CompanyD |
| 100005 | sweaters | 194 | 24h | 21009 | Accesorios | 19.90 | blue | NA | 110005 | CompanyE |

**products_with_supplier_contacts:** This view presents product information along with supplier contact details, including the supplier's name and postal code.

Statement:

```
CREATE VIEW products_with_supplier_contacts AS
SELECT p.Sku_id, p.Type, p.Unit_price, p.Color, p.Size,
       s.Supplier_id, s.Name AS Supplier_name, CONCAT(s.First_name, ' ', s.Last_name) AS Supplier_contact_name, s.S_Postal_code AS Supplier_postal_code
FROM Product p
JOIN Supplier_has_Product shp ON p.Sku_id = shp.Product_Sku_id
JOIN Supplier s ON shp.S_Supplier_id = s.Supplier_id;
SELECT * FROM products_with_supplier_contacts;
```

Instance:

Result Grid Filter Rows: Search Export:

| Sku_id | Type | Unit_price | Color | Size | Supplier_id | Supplier_name | Supplier_contact_na... | Supplier_postal_co... |
|---|---|---|---|---|---|---|---|---|
| 10771 | Kid | 29.90 | white | S | 110004 | CompanyD | Lock Zhu | M1E 3B3 |
| 14623 | Men | 39.90 | blue | M | 110001 | CompanyA | Ju Mie | M1M 3A3 |
| 21009 | Accesorios | 19.90 | blue | NA | 110005 | CompanyE | Ivey Western | M1S 3A4 |
| 21769 | Men | 39.90 | black | M | 110002 | CompanyB | Juliet Mia | M1S 3A3 |
| 47123 | Women | 39.90 | green | S | 110003 | CompanyC | Key Zhu | M1T 3L3 |

**suppliers_with_addresses_and_billing:** This view displays supplier information, including their addresses and billing details.

Statement:

```
CREATE VIEW suppliers_with_addresses_and_billing AS
SELECT s.Supplier_id, s.Name AS Supplier_name,
       a.Street AS Address_street, a.City AS Address_city, a.Province AS Address_province, a.Country AS Address_country,
       a.Street AS Billing_street, a.City AS Billing_city, a.Province AS Billing_province, a.Country AS Billing_country
FROM Supplier s
JOIN Address a ON s.S_Postal_code = a.Postal_Code;
SELECT * FROM suppliers_with_addresses_and_billing;
```

Instance:

Result Grid Filter Rows: Search Export:

| Supplier_id | Supplier_name | Address_street | Address_city | Address_provin... | Address_count... | Billing_street | Billing_ci... | Billing_provin... | Billing_coun... |
|---|---|---|---|---|---|---|---|---|---|
| 110001 | CompanyA | 1 Yonge Street | Toronto | ON | Canada | 1 Yonge Street | Toronto | ON | Canada |
| 110002 | CompanyB | 12 Yonge Street | Toronto | ON | Canada | 12 Yonge Street | Toronto | ON | Canada |
| 110003 | CompanyC | 13 Yonge Street | Toronto | ON | Canada | 13 Yonge Street | Toronto | ON | Canada |
| 110004 | CompanyD | 145 Yonge Street | Toronto | ON | Canada | 145 Yonge Street | Toronto | ON | Canada |
| 110005 | CompanyE | 167 Yonge Street | Toronto | ON | Canada | 167 Yonge Street | Toronto | ON | Canada |

## Queries(SQL statements and instance)

In the SHEIN database, ten SQL queries have been executed to efficiently access and display data from multiple tables. These queries aggregate, filter, and format the information for specific purposes. We will describe each of them in the following:

Top 5 products with the highest unit prices: This query retrieves the product details of the top 5 products with the highest unit prices.

Statement:

```
-- find the top 5 products with the highest unit prices
SELECT *
FROM Product
ORDER BY Unit_price DESC
LIMIT 5;
```

Instance:

Result Grid Filter Rows: Search Edit: Export/Import:

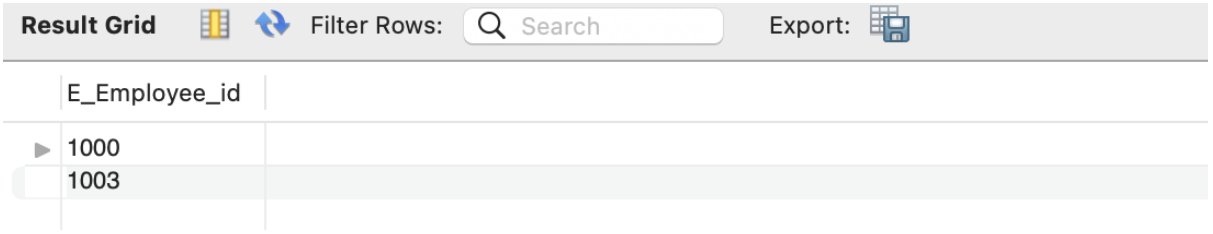
| Sku_id | Type | Color | Size | Unit_price |
|---|---|---|---|---|
| 14623 | Men | blue | M | 39.90 |
| 21769 | Men | black | M | 39.90 |
| 47123 | Women | green | S | 39.90 |
| 10771 | Kid | white | S | 29.90 |
| 21009 | Accesorios | blue | NA | 19.90 |
| NULL | NULL | NULL | NULL | NULL |

Employees with more than one permission: This query identifies employees who have been granted more than one permission in the system.

Statement:

```
-- find which employee has more than one permission
SELECT E_Employee_id
FROM Employee_Permission
GROUP BY E_Employee_id
HAVING COUNT(*) > 1;
```

Instance:

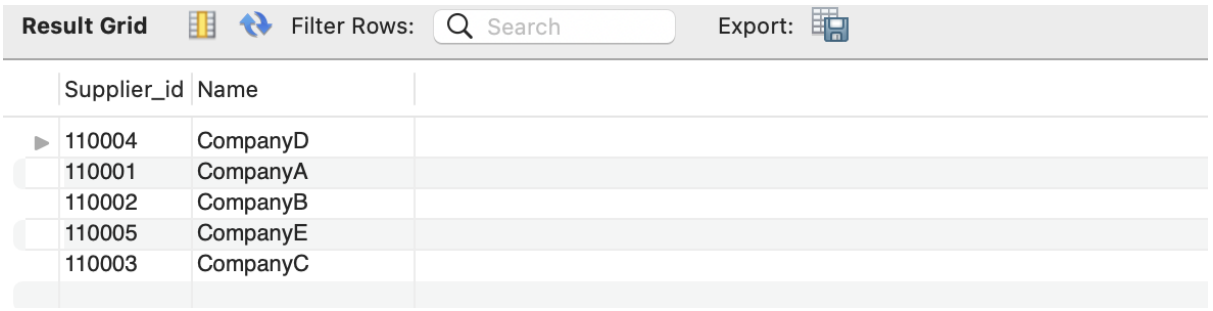



Suppliers located on Yonge Street: This query lists the supplier IDs and names of suppliers whose address is on Yonge Street.

Statement:

```
-- find suppliers whose address is on Yonge Street
SELECT s.Supplier_id, s.Name
FROM Supplier s
INNER JOIN Address a ON s.S_Postal_code = a.postal_code
WHERE a.Street LIKE '%Yonge%';
```

Instance:

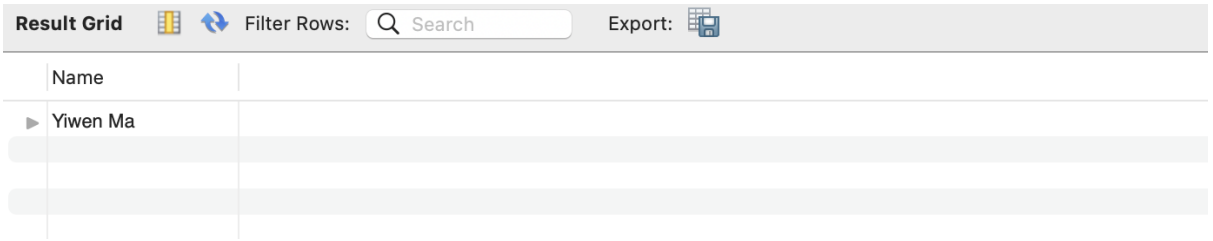



Employee who made order 100002: This query retrieves the name of the employee responsible for creating purchase order 100002.

Statement:

```
-- Find who make the order 100002?
SELECT e.Name
FROM Employee e
JOIN Employee_makes_Purchase_Order empo ON e.Employee_id = empo.E_Employee_id
JOIN Purchase_Order po ON po.Order_id = empo.P_Order_Order_id
WHERE po.Order_id = 100002;
```

Instance:

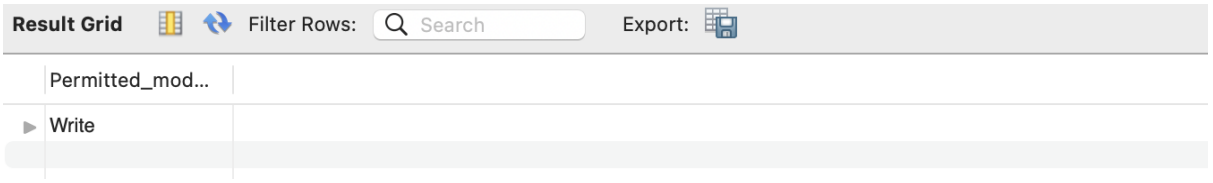



Yiwen Ma's permission module: This query finds the permission module(s) assigned to employee Yiwen Ma.

Statement:

```
--  Find employee Yiwen Ma's permission module?
SELECT ep.Permitted_module
FROM Employee_Permission ep
JOIN Employee e ON ep.E_Employee_id = e.Employee_id
WHERE e.Name = 'Yiwen Ma';
```

Instance:

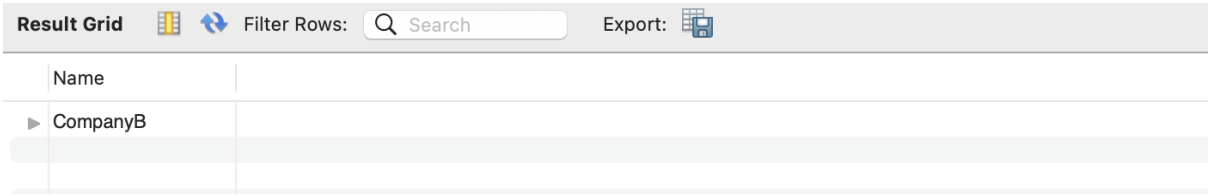



Supplier of product sku21769: This query identifies the supplier name for product sku21769.

Statement:

```
-- Find sku21769 is made by which supplier?
SELECT s.Name
FROM Supplier s
JOIN Supplier_has_Product shp ON s.Supplier_id = shp.S_Supplier_id
JOIN Product p ON shp.Product_Sku_id = p.Sku_id
WHERE p.Sku_id = 21769;
```

Instance:



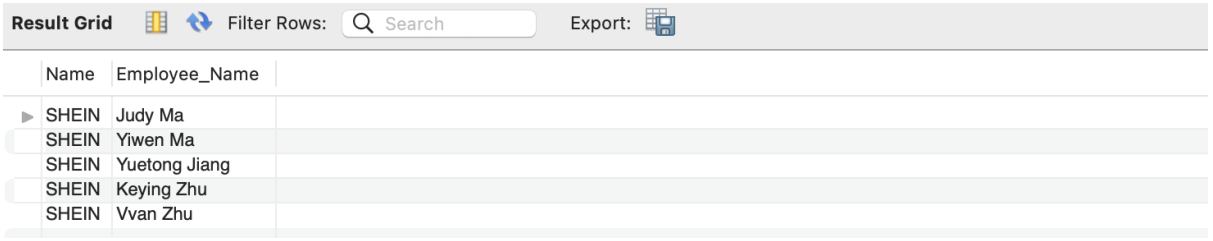

| Name |
|---|
| CompanyB |

Company names and associated employees: This query lists the names of all companies and their corresponding employees.

Statement:

```
-- Find the names of all companies and their associated employees
SELECT c.Name, e.Name AS Employee_Name
FROM `Company-SHEIN` c
RIGHT JOIN Employee e ON c.Company_id = e.C_id;
```

Instance:



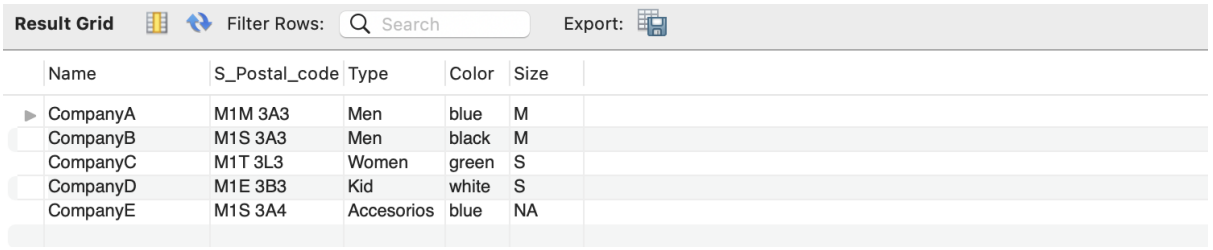

| Name | Employee_Name |
|---|---|
| SHEIN | Judy Ma |
| SHEIN | Yiwen Ma |
| SHEIN | Yuetong Jiang |
| SHEIN | Keying Zhu |
| SHEIN | Vvan Zhu |

Supplier names, postal codes, and associated products: This query retrieves supplier names, postal codes, and the products they supply.

Statement:

```
-- Find the names and postal codes of all suppliers and their associated products
SELECT s.Name, s.S_Postal_code, p.Type, p.Color, p.Size
FROM Supplier s
INNER JOIN Supplier_has_Product shp ON s.Supplier_id = shp.S_Supplier_id
INNER JOIN Product p ON shp.Product_Sku_id = p.Sku_id;
```

Instance:



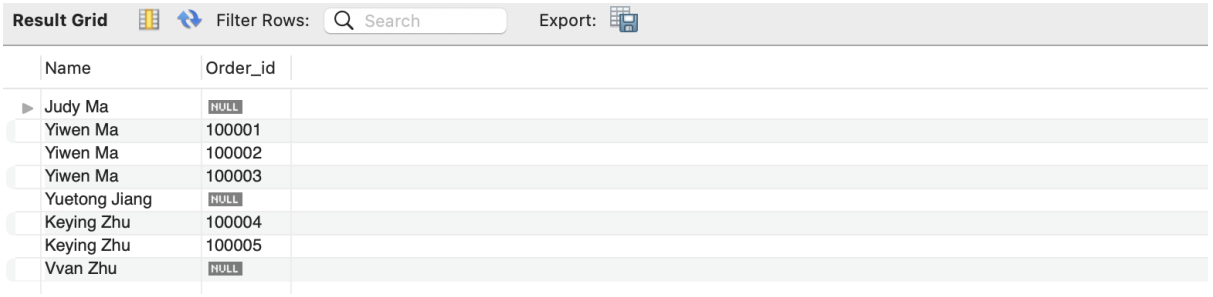

| Name | S_Postal_code | Type | Color | Size |
|---|---|---|---|---|
| CompanyA | M1M 3A3 | Men | blue | M |
| CompanyB | M1S 3A3 | Men | black | M |
| CompanyC | M1T 3L3 | Women | green | S |
| CompanyD | M1E 3B3 | Kid | white | S |
| CompanyE | M1S 3A4 | Accesorios | blue | NA |

Employees and their associated purchase orders: This query lists all employees along with their related purchase orders.

Statement:

```
-- Find all employees and their associated purchase orders
SELECT e.Name, po.Order_id
FROM Employee e
LEFT JOIN Employee_makes_Purchase_Order empo ON e.Employee_id = empo.E_Employee_id
LEFT JOIN Purchase_Order po ON empo.P_Order_Order_id = po.Order_id;
```

Instance:



Result Grid Filter Rows: Search Export:

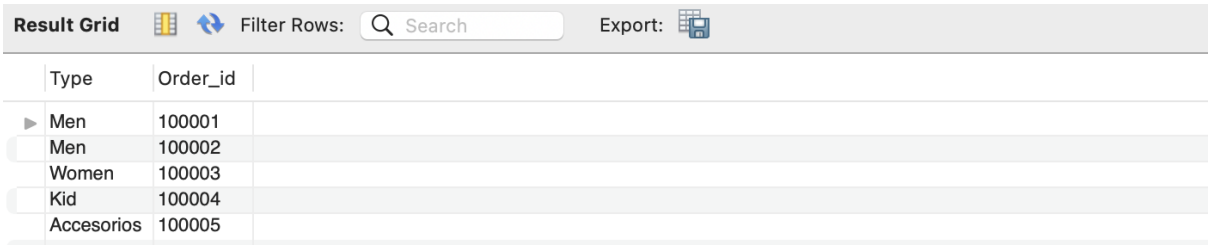
| Name | Order_id |
|---|---|
| Judy Ma | NULL |
| Yiwen Ma | 100001 |
| Yiwen Ma | 100002 |
| Yiwen Ma | 100003 |
| Yuetong Jiang | NULL |
| Keying Zhu | 100004 |
| Keying Zhu | 100005 |
| Vvan Zhu | NULL |

Products and their associated purchase orders: This query displays all products and the purchase orders in which they are included.

Statement:

```
-- Find all products and their associated purchase orders
SELECT pr.Type, po.Order_id
FROM Product pr
RIGHT JOIN Purchase_Order_Contains_Product pop ON pr.Sku_id = pop.P_Sku_id
RIGHT JOIN Purchase_Order po ON pop.P_Order_id = po.Order_id;
```

Instance:



Result Grid Filter Rows: Search Export:

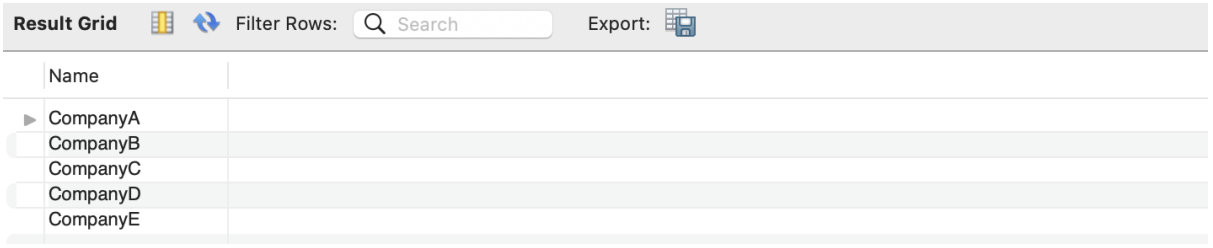
| Type | Order_id |
|---|---|
| Men | 100001 |
| Men | 100002 |
| Women | 100003 |
| Kid | 100004 |
| Accesorios | 100005 |

Suppliers who have not made any payments: This query identifies suppliers who have not made any payments and lists their names.

Statement:

```
-- Find all suppliers who have not made any payments, and list their names
SELECT s.Name
FROM Supplier s
LEFT JOIN Payment p ON s.Supplier_id = p.C_Company_id
WHERE p.Transaction_id IS NULL;
```

Instance:




## Triggers (SQL statements and instance)

### Trigger 1: prevent_CEO_deletion

This trigger safeguards against the accidental or unauthorized deletion of a CEO from the Employee table by raising an exception and displaying a custom error message when an attempt is made to delete a row with a 'CEO' title.

```
331    DELIMITER $$
332
333 •  CREATE TRIGGER prevent_CEO_deletion
334      BEFORE DELETE
335      ON Employee
336      FOR EACH ROW
337    BEGIN
338      IF OLD.title = 'CEO' THEN
339        SIGNAL SQLSTATE '45000'
340        SET MESSAGE_TEXT = 'Cannot delete a CEO';
341      END IF;
342    END$$
343
344    DELIMITER ;
```

Upon successfully creating the trigger, we proceeded to test its functionality using a query.

```
346 •  DELETE FROM Employee
347    WHERE Employee_id = 1003;
```

The obtained result demonstrates that the trigger is functioning as intended.




### Trigger 2: purchase_order_quantity_check

This trigger ensures that every purchase order inserted into the Purchase_Order table has a minimum quantity of 5 items. If the quantity is less than 5, the insertion will be blocked, and an error message will be displayed

```
351    DELIMITER $$
352 •  CREATE TRIGGER purchase_order_quantity_check
353    BEFORE INSERT ON Purchase_Order
354    FOR EACH ROW
355    BEGIN
356      IF NEW.quantity < 5 THEN
357        SIGNAL SQLSTATE '45000'
358          SET MESSAGE_TEXT = 'Error: Quantity is too low';
359      END IF;
360    END; $$
361    DELIMITER ;
```

Upon successfully creating the trigger, we proceeded to test its functionality using a query.

```
363    INSERT INTO Purchase_Order (Order_id, Delivery_time, Quantity, Order_detail)
364    VALUES (100006, '24h', 1, 'shirts')
```

The obtained result demonstrates that the trigger is functioning as intended.




### Trigger 3: Unit_Price_Check

This trigger helps to prevent updates that would result in a higher unit price than the previous value, ensuring that any price increase is restricted or controlled as per the business logic.

```
DELIMITER $$
CREATE TRIGGER Unit_Price_Check
BEFORE UPDATE ON Product
FOR EACH ROW
BEGIN
  IF NEW.Unit_price > OLD.Unit_price THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Error: The unit price is too high';
  END IF;
END; $$
DELIMITER ;
```

Upon successfully creating the trigger, we proceeded to test its functionality using a query.

```
UPDATE Product
SET Unit_price = 100
WHERE Sku_id = 14623;
```

The obtained result demonstrates that the trigger is functioning as intended.

```
43  15:07:48  UPDATE Product SET Unit_price = 100 WHERE Sku_id = 14623  Error Code: 1644. Error: The unit price is too high
```

# Database Catalog

## Catalog

### Relations

| Relation_name | No_of_Columns | Keys | | |
|---|---|---|---|---|
| | | Primary | Foreign | Candidate Key |
| Company | 4 | Company_id | | C_Postal_code |
| Supplier | 9 | Supplier_id | P_Transaction_id | Postal_Code |
| Product | 5 | Sku_id | | |
| Payment | 6 | Transaction_id | C_Company_id | P_Postal_code |
| Purchase_order | 4 | Order_id | | |
| Employee | 7 | Employee_id | C_id | |
| Employee_Login | 3 | Employee_Login_id | Em_Employee_id | |
| Employee_permission | 3 | Permission_id | Employee_id | |
| Purchase_Order_Contains_Product | 2 | | P_Order_id<br>P_Sku_id | |
| Employee_Makes_Purchase Order | 2 | | E_Employee_id<br>P_Order_id | |
| Supplier_Produces_Product | 2 | | P_Sku_id<br>S_Supplier_id | |
| Address | 5 | Postal_code | | |

#### Columns

| Column Name | Entity | Type of Attribute | Data_type | Domains and Constraints | Description |
|---|---|---|---|---|---|
| Company_id | Company | Simple | Int | | represents a unique identifier assigned to each company in the database, for this case, only represent one company, SHEIN |
| Name | Company | Simple | VARCHAR(45) | Default = Shein | Stores the name of company, for this case, default would be SHEIN |
| Contact Information | Company | Simple | VARCHAR(45) | | stores the contact details (e.g., phone number, email address) of the company |
| C_Postal_code | Company | Simple | VARCHAR(7) | Contains multiple data elements: {Street, City, Province, Country, Postal_code} | Stores the postal code of the company |

| | | | | | |
|---|---|---|---|---|---|
| Supplier_id | Supplier | Simple | Int | Not Null | contains a unique identifier for each supplier |
| Name | Supplier | Simple | VARCHAR(45) | Not Null | Stores the name of the supplier |
| Type | Supplier | Simple | VARCHAR(20) | One of {Sole Proprietorship, Partnership, Corporation, and S Corporation} | Store the type of business of a suppiler |
| Bank_name | Supplier | Simple | VARCHAR(45) | Not Null | Store the the name of the bank that supplier opened his/her account with |
| Account_number | Supplier | Simple | VARCHAR(45) | Not Null | Store the name of the bank that supplier opened his/her account with |
| First_name | Supplier | Simple | VARCHAR(45) | Not Null | Store the first name of the bank that supplier opened his/her account with |
| Last_name | Supplier | Simple | VARCHAR(45) | Not Null | Store the last name of the bank that supplier opened his/her account with |
| Postal_Code | Supplier | Simple | VARCHAR(7) | Format:{A1A 1A1} | serves as the candidate key identifying each address record of supplier |
| P_Transaction_id | Supplier | Simple | Int | Not Null | serves as a foreign key that uniquely identifies each payment transaction made by SHEIN to the supplier |
| Sku_id | Product | Simple | Int | Not Null | represents the unique identifier for each product stock-keeping unit (SKU) and stores the information accordingly |
| Type | Product | Simple | VARCHAR(10) | Choose from {Men, Women, Oversized, Kid, Accesorios, Other} | stores the product type that choose from {Men, Women, Oversized, Kid, Accesorios, Other} for a particular product |
| Unit_price | Product | Simple | Decimal(6,2) | Non-Negative | represents and stores the price of a single unit of the product |
| Color | Product | Simple | VARCHAR(45) | | represents and stores the color information of the product |
| Size | Product | Simple | VARCHAR(10) | Choose from {XS,S,M,L,XL,XLL,XLLL} | stores the size information that choose from {XS,S,M,L,XL,XLL,XLLL}for a particular product |
| Transcation_id | Payment | Simple | Int | Not Null | uniquely identifies each payment transaction made by SHEIN to the supplier |
| Payment_amount | Payment | Simple | Decimal(10,2) | Non-Negative | stores the amount paid by SHEIN for a transaction to the supplier |
| Payment_type | Payment | Simple | VARCHAR(45) | | stores the payment by SHEIN for a transaction to the supplier |
| Transaction_date | Payment | Simple | Datetime | Prior to the current time | indicates and stores the date when a payment was made for a particular transaction to supplier |
| P_Postal_code | Payment | Simple | VARCHAR(7) | Format:{A1A 1A1} | serves as the foreign key identifying each unique address record of supplier |
| C_Company_id | Payment | Simple | Int | Not Null | serves as a foreign key that assigned to each company in the database, for this case, only represent one company, SHEIN |

| | | | | | |
|---|---|---|---|---|---|
| Order_id | Purchase_Order | Simple | Int | Not Null | stores a unique identifier for each purchase order placed by an employee |
| Order_detail | Purchase_Order | Simple | VARCHAR(45) | | stores order information related to a specific order |
| Quantity | Purchase_Order | Simple | Int | Positive | stores total amount of quantity of a purchase order |
| Delivery_time | Purchase_Order | Simple | VARCHAR(5) | One of {24hr, 48hr, 5days, 7days} | stores information on when the order should be deliverid in each order, should choose from {24hr, 48hr, 5days, 7days} |
| Employee_id | Employee | Simple | Int | Not Null | stores identifies a unique employee within the organization - SHEIN |
| Name | Employee | Simple | VARCHAR(30) | Not Null | stores the full name of the employee |
| Email | Employee | Simple | VARCHAR(45) | Not Null, Unique | stores the company email of the employee |
| Mobile | Employee | Simple | VARCHAR(15) | Not Null, Unique | stores the mobile number of the employee |
| Title | Employee | Simple | VARCHAR(45) | | stores the mposition/title of the employee |
| C_id | Employee | Simple | Int | Not Null | serves as foreign key that represents a unique identifier assigned to each company in the database, for this case, only represent one company, SHEIN |

| | | | | | |
|---|---|---|---|---|---|
| Employee_Login_id | Employee_Login | Simple | Int | Not Null | stores unique login IDs for each employee to access the company's information systems. |
| Em_Employee_id | Employee_Login | Simple | Int | Not Null | serves as foreign key with unique employee id within the organization - SHEIN |
| Password | Employee_Login | Simple | VARCHAR(45) | Length >= 8; Contains at least one of each: Alphabet, Number, Special Char | stores the encrypted passwords of employees for secure authentication purposes to log in |
| Permission_id | Employee_Permission | Simple | Int | Not Null | stores the the unique identifier for the level of access and permissions granted to an employee |
| Permitted_module | Employee_Permission | Simple | VARCHAR(45) | One of {Read, Write or Execute} | stores information on specific permitted module granted to an employee, should choose from Read, Write or Execute} |
| Employee_id | Employee_Permission | Simple | Int | Not Null | serves as foreign key with unique employee id within the organization - SHEIN |
| P_Order_id | Purchase_Order_Contains_Product | Simple | Int | Not Null | serves as a foreign key that stores a unique identifier for each purchase order placed by an employee |
| P_Sku_id | Purchase_Order_Contains_Product | Simple | Int | Not Null | serves as a foreign key that represents the unique identifier for each product stock-keeping unit (SKU) |
| E_Employee_id | Employee_Makes_Purchase Order | Simple | Int | Not Null | serves as a foreign key that represents the unique employee id within the organization - SHEIN |
| P_Order_id | Employee_Makes_Purchase Order | Simple | Int | Not Null | serves as a foreign key that represents unique identifiers for each purchase order placed by an employee |
| P_Sku_id | Supplier_Produces_Product | Simple | Int | Not Null | serves as a foreign key that represents the unique identifier for each product stock-keeping unit (SKU) |
| S_Supplier_id | Supplier_Produces_Product | Simple | Int | Not Null | serves as a foreign key that represents the unique identifier for each supplier |
| Postal_code | Address | Simple | VARCHAR(7) | Format:{A1A 1A1} | serves as the primary key identifying each unique address record of supplier |
| Country | Address | Simple | VARCHAR(45) | | stores country information of a supplier |
| Province | Address | Simple | VARCHAR(45) | | stores province information of a supplier |
| City | Address | Simple | VARCHAR(45) | | stores city information of a supplier |
| Street | Address | Simple | VARCHAR(45) | | stores street information of a supplier |

## Discussion on Legal, Ethical & Security Aspects

Ensuring adherence to legal, ethical, and security regulations is of utmost importance in data management and processing. With the increasing amount of data collected and stored, there is a greater need for businesses and organizations to ensure that these standards are followed. Legal practices refer to how organizations adhere to the rules and regulations that govern data gathering, administration, and utilization, and failure to comply usually results in legal implications such as fines, legal action, and reputational harm to the organization. Moreover, ethical practices are also crucial when it comes to data handling. Organizations should ensure that customer data that is collected should be used in an ethical manner and avoid practices that could potentially lead to harm or discrimination against individuals. Moreover, security practices are essential for protecting customer and personal data from threats such as hacking, data breaches, and cyber-attacks which may result in severe harm to users and companies. It is crucial for organizations to prioritize legal, ethical, and security aspects of data handling to ensure their business's longevity. The following sections will go over each aspect in depth in order to examine some current practices and offer suggestions on how SHEIN might adopt these strategies to its own use.

## Legal Aspects

Legal practices are critical in ensuring that companies follow the rules and regulations governing data use. In Canada, there are various laws that control the cybersecurity legal environment, including privacy, anti-spam, criminal responsibility, and intellectual property (Mckenzie, 2022). In general, there are federal and provincial laws regulating how personal information can be collected, used, or disclosed in Canada.

**Federal - Personal Information Protection and Electronic Documents Act (PIPEDA)**

The Personal Information Protection and Electronic Documents Act (PIPEDA) is an Act that supports and promotes electronic commerce by protecting personal information collected, used, or disclosed in certain circumstances (Government of Canada, 2019). By allowing the use of electronic means to communicate or record information or transactions, and by amending the Canada Evidence Act, the Statutory Instruments Act, and the Statute Revision Act (Government of Canada, 2019).

The Personal Information Protection and Electronic Documents Act (PIPEDA) federal law applies to private sectors that engage in collecting, using, or disclosing personal information in the course of their commercial operations (Government of Canada, 2019). This Act also establishes regulations for the data collection, management and usage of personal information which requires organizations to seek individual consent before collecting, using, or disclosing personal information about them, and it also requires organizations to secure personal information by applying suitable security measures. Furthermore, PIPEDA grants individuals the right to view and amend any mistakes in their personal information collected and stored by an organization. People may also file a complaint with the Office of the Privacy Commissioner of Canada if they find out an organization has infringed their PIPEDA privacy rights.

One critical aspect should be highlighted that PIPEDA only applies to the private sector and does not include the personal information collected, used, or disclosed by federal or provincial governments (Government of Canada, 2019). In addition, some jurisdictions, such as Alberta's Personal Information Protection Act (PIPA) and British Columbia's Personal Information Protection Act (PIPA), have their own privacy standards that apply to firms operating inside their regions (PIPABC). Moreover, if a firm fails to adhere to the Personal Information Protection and Electronic Documents Act (PIPEDA), severe legal and financial implications will occur. The Office of the Privacy Commissioner of Canada (OPC) is in charge of implementing PIPEDA, and it has the authority to investigate the case and take enforcement action against organizations that breach the law. Thus, it is critical for organizations to adhere and follow the PIPEDA responsibilities seriously and employ suitable data security measures to collect, use and disclose personal information in the right manner.

**Provincial - Alberta - Personal Information Protection Act, SA 2003, c P-6.5 (Alberta PIPA)**

The Personal Information Protection Act (PIPA) is Alberta’s private sector privacy law and operates in a different scope compared to federal PIPEDA. PIPA regulates private sector organizations, businesses, and in some cases, non-profit organizations as well in the Alberta region that aim to protect personal information and grant the right of access to an individual’s personal information (Government of Alberta, 2023). Organizations subject to PIPA must design and implement appropriate policies in order to satisfy their statutory requirements. When PIPA refers to anything or something as 'reasonable’, it means that it is something that a reasonable person would consider appropriate in the circumstances (Government of Alberta, 2023). PIPA was presented as Bill 44 in the Alberta Legislature on May 14, 2003, and went into effect on

January 1, 2004. Since then, it has been amended four times and the last amendment was on December 17, 2014. In this amendment, this Act changes PIPA to authorize a trade union to collect, utilize, and disclose personal information without consent in order to inform or convince the public about a labor relations issue that is of major public interest or significance. On December 17, 2014, the modifications got Royal Assent and went into force immediately (Government of Alberta, 2023). Unlike PIPEDA, PIPEDA requires companies to notify any breach of security protections involving personal information that creates a risk of serious damage to persons to the Office of the Privacy Commissioner of Canada and impacted individuals. The Alberta PIPA compels organizations to notify breaches to impacted persons as well, however, the reporting level is significantly different.

While PIPEDA and Alberta PIPA have many parallels in terms of regulating the processing of personal information by private sector entities, there are also significant variances in scope and standards. Entities operating in Alberta must follow the Alberta PIPA, whereas those operating elsewhere in Canada (with a few exceptions) must follow the PIPEDA (Government of Alberta, 2023).

**Proposed - Artificial Intelligence and Data Act (AIDA)**

There is a proposed Act that has been discussed a lot recently across Canada, the Artificial Intelligence and Data Act (AIDA). In April 2021, Bill C-11 the Digital Charter Implementation Act was introduced by the Canadian Minister of Innovation, Science, and Industry. The law incorporates AI and automated decision-making provisions that compel businesses to be transparent about their use of AI and to provide explanations for decisions made by automated systems. Bill C-27, the evolution version of C-11, was submitted in June 2022 as an Act to establish the Consumer Privacy Protection Act, the Personal Information and Data Protection Tribunal Act, and the Artificial Intelligence and Data Act (AIDA) (Mckenzie, 2022). Bill C-27 was presented to improve the framework for the protection of personal information in the private sector by overhauling PIPEDA. Bill C-27 is now being debated in Parliament and, if enacted, would introduce the new measures in AIDA (Mckenzie, 2022). The new rules suggested aim to regulate international and interprovincial trade as well as business in artificial intelligence systems. AIDA would provide uniform standards for the design, development, and use of artificial intelligence systems, including safeguards against harm and biased output. AIDA would also forbid some activities involving data and artificial intelligence systems that might cause substantial harm to persons or their interests (Mckenzie, 2022).

**Conclusion**

SHEIN is one of the leading online retailers in the world that collects, uses, and stores a significant amount of personal information from its suppliers and customers in Canada and other countries SHEIN operates in. This information includes but is not limited to, names, addresses, phone numbers, payment information, and so on. As a result, SHEIN is required to follow the Personal Information Protection and Electronic Documents Act (PIPEDA), Canada's federal privacy law that governs how private sector businesses acquire, use, and disclose personal information in the course of their commercial activities. Moreover, as mentioned in the above section, the Artificial Intelligence and Data Act (AIDA) was proposed in 2022 and if this act

passes, SHEIN as a company heavily leveraged by Artificial intelligence will need to follow the new act accordingly. There are several reasons why it is important to follow and adhere to a country's law on data collection, management, and usage. Firstly, failure to comply with PIPEDA can result in hefty financial penalties and legal consequences. Anyone harmed by a violation of privacy may also seek damages in court. Secondly, compliance with PIPEDA is critical to establishing and retaining brand trustworthiness with its suppliers and customers. SHEIN may develop a great brand image and increase cooperative partner loyalty by displaying a dedication to preserving personal information that has been collected. In conclusion, by adhering to and prioritizing data privacy compliance, SHEIN may establish a solid foundation for its success in Canada while also demonstrating a dedication to protecting its supplier and customer information.

**Ethical Aspects**

The moral principles and ideals that direct how individuals and organizations ought to conduct themselves are referred to as ethics. When creating a database, it is essential to give careful attention to ethical considerations, as doing so helps to guarantee the responsible and respectful handling of sensitive information, cultivates confidence among stakeholders, and ensures continued compliance with applicable laws and regulations. When it comes to our business, ethical considerations include things like maintaining the trust of our customers, being transparent, adhering to fair labor standards, maintaining a sustainable environment, and handling customer data responsibly. Taking action to address these ethical concerns demonstrates SHEIN's dedication to operating its business in a way that is respectful of the rights and interests of all of its stakeholders, which includes its consumers, employees, and suppliers, as well as the environment.

SHEIN needs to resolve not only concerns regarding the protection of its customers' personal information and privacy, but also concerns regarding the ethical conduct of its business operations. In the following section, we will elucidate on additional ethical considerations, such as non-deceptive business practices, equitable treatment of customers, and the social responsibility of corporations. (Bhattacharya et al., 2021). Increased Non-Deceptive Methods refers to the enhancement of transparent and honest practices in an online retail environment. These methods involve providing accurate product information, clear pricing, honest marketing, genuine customer feedback, and easily accessible policies.(Grewal, 2020). Equitable Treatment of Customers refers to the fair and just treatment of all customers by a business, ensuring that their needs are met without bias or discrimination. This involves providing equal access to products, services, and support, as well as addressing concerns and resolving issues promptly and fairly. Corporate Social Responsibility (CSR) refers to the commitment of businesses to conduct their operations in an ethical, socially responsible, and environmentally sustainable manner. (Ogunmola & Kumar, 2020). In addition to this, a very important point is data privacy training, In the digital age of today, where data intrusions and cyberattacks are on the rise, data privacy employee training is crucial. Training employees on data privacy helps them comprehend their role in preserving personal data and best practices for doing so. This includes understanding how sensitive data is managed and stored, identifying potential threats such as phishing or malware, and knowing what to do in the event of a breach. Regularly training employees on data privacy can help organizations cultivate a culture that prioritizes security and mitigates the risk of data

breaches. This not only protects the company from potential legal and financial repercussions, but it also serves to preserve the confidence of customers and other stakeholders.

**Amazon - Code of Business Conduct and Ethics**
As one of the world's largest e-commerce platforms, Amazon has developed a comprehensive strategy for addressing ethical concerns regarding privacy and personal data. According to Amazon's Code of Business Conduct and Ethics, The following are the main aspects they mentioned:

**Ethical use of data and artificial intelligence:** Amazon uses customer data to improve its services and personalize experiences, such as product recommendations and targeted advertising (n.d.). The company is committed to using data responsibly and ensuring that its artificial intelligence systems are designed and implemented with ethical considerations in mind(n.d.). This includes avoiding bias in algorithms and promoting fairness, transparency and accountability in AI applications.

**Privacy policy:** Amazon has a detailed privacy policy that explains how they collect, use, and share customer information (n.d.). This policy covers the types of information collected (e.g., personal information, browsing data, device information), the purposes for data collection (e.g. , order processing, personalized recommendations), and the circumstances under which data may be shared with third parties (e.g., sellers, service providers).

**User control over personal data:** Amazon provides customers with tools to manage their privacy settings, review their data, and exercise their data rights (n.d.). For example, customers can access and edit their personal information, control marketing communications, and request data deletion or portability. Amazon also offers a "Privacy Dashboard" that allows customers to review and adjust their privacy settings easily.

**Transparency and communication:** Amazon takes transparent communication with customers about their privacy practices very seriously. They proactively notify customers of any updates or changes to their privacy policies through timely notifications. Amazon also details how customer data is collected, processed and used for various purposes, such as generating personalized recommendations or targeted advertising.

**Conclusion - Establish code of ethics**
SHEIN should develop a comprehensive code of ethics that is tailored to its particular demands and industry standards.To ensure ethical data practices, SHEIN can learn from industry best practices, including Amazon's and other e-commerce titans' adherence to certain guidelines. The structure of this policy should incorporate privacy, personal data, and other pertinent ethical concerns in order to foster consumer trust, improve brand reputation, and ensure responsible data management practices within the organization.

SHEIN can further improve the company's reputation in the market and win the trust of customers by formulating a code of ethics, as a clearly defined code of ethics demonstrates SHEIN's commitment to ethical practices and responsible data management, fostering trust among customers and stakeholders. In addition, the development of ethical guidelines helps to ensure that SHEIN complies with relevant data protection regulations, which can reduce the risk of legal penalties, fines or sanctions. The process of developing a code of ethics should ensure that a comprehensive code of ethics provides employees with clear guidance on how to handle customer

data responsibly, minimizing the potential for misuse or mishandling of sensitive information. Many companies prioritize data privacy employee training to ensure that their employees understand the importance of protecting sensitive data. For example, Google provides regular security and privacy training to its employees, covering topics such as phishing, password security, and data handling best practices. It is worth noting that a code of ethics should create a consistent and standardized approach, thereby reducing confusion and promoting a common understanding of a company's ethical principles. Finally, from a social perspective, establishing a code of ethics reflects SHEIN's commitment to corporate social responsibility, showing customers, partners and investors that the company values ethical behavior and takes its social and environmental impacts seriously.

## Security Aspects

In today's digital age, it is more important than ever to prioritize data security when working with a database, as the database can hold vast amounts of sensitive information, including personal information and confidential business records, which, if leaked, can lead to serious consequences. For instance, in an SRM (Supplier Relationship Management) database, there could be data records related to employee personal contacts, supplier banking information, and modules related to the database or system access and permissions. It is, therefore, essential to maintain a secure and confidential database with limited access to different groups. As the goal of the attackers is to exploit a vulnerability by treating the system in some undesirable ways, we should mitigate and reduce the attack surface, which means any possible exploit that may pose a threat to the system (Riley, 2015). Aside from physical security (securing the hardware devices and machines that connect to the database physically), in this section, we will discuss two critical aspects of database security: confidentiality and integrity. We will explore practical methods and strategies that can be used to ensure that the database and the information it stores are well-protected. For example, applying the data security measures such as access control, data encryption, and regular auditing and backing up, organizations can ensure that sensitive data is protected from tampering, data leakage, and unauthorized access and disclosure.

## Data Confidentiality

To ensure data confidentiality, it is crucial to restrict access to the database and the information stored only to authorized persons or devices. Unauthorized access poses a significant threat to data security and must be prevented at all costs. However, access restrictions alone are not sufficient to ensure database security, as the availability of the data is equally important (Riley, 2015). Any disruption or malfunction in the database can leave the data vulnerable to attacks. Moreover, data privacy is closely tied to data security, as it is necessary to determine what data is considered to be confidential and should be private and limit access accordingly. This raises ethical questions regarding the protection of sensitive information. In most cases, data that is relevant to other parties' interests should be treated with the utmost caution and require permission or consent for disclosure. Proper management of data privacy and confidentiality is critical to maintain trust with customers and clients, and failure to do so can have severe legal and financial implications for an organization. Therefore, strategies must be put in place to prevent potential data leaks during database operations.

One effective way to protect the system against malware and viruses is through the use of firewalls and antivirus software. These tools are specifically designed to check the system for potential viruses, and protect the computer from malware and viruses that arrive through the network. Another way to defend against data breaches is through data encryption, which scrambles confidential information into a ciphertext that can only be understood by authorized parties with specific decryption keys. Furthermore, the system access control by giving proper authorization to the proper group can help maintain data confidentiality. People should have unique login identifiers for them to access to the database, which are associated with certain permissions and authorizations. In our SRM system, we have included two tables as examples to illustrate the access control practice. One table contains data related to employee login, and the other contains employee's permission data, where the multiple permitted modules such as read, write, execute are assigned to each employee. Triggers can be designed to control the access and operations within the database. For example, employees with read-only permission cannot alter any data records or place purchase orders or make payments, with the use of update or insert statements. Triggers designed before 'Update', 'Insert', 'Delete' can capture unauthorized actions and stop the operation, or even rollback the original data record. Additionally, triggers can also be used for auditing and monitoring purposes by tracking any user activities and recording all access into a specific audit table as a kind of log file. Performing routine log reviews is essential to promptly detect and address any potential security incidents. Besides, views in SQL are virtual tables that serve the purpose of not only simplifying the complex and repetitive procedures of querying data from multiple tables, but also controlling the user access as an encapsulation. Instead of granting permission to the whole system or all tables, views allow for the assignment of limited access to a certain amount of data stored in the system. This limits users or clients from accessing the entire table or underlying data, effectively protecting confidential data and achieving data security. By implementing these measures, organizations can safeguard their sensitive data and prevent unauthorized access or tampering.

**Data Integrity**

Data integrity is a fundamental aspect that involves ensuring that the data stored in a database is reliable and consistent, with no unauthorized or unintended changes or corruption. Maintaining data integrity is essential for preserving the trustworthiness and reliability of data, avoiding data breaches, errors, or data leaks. There are various techniques that can be applied to ensure that data integrity is maintained throughout the entire lifecycle of a database, from database creation, data insertion, to updating and deletion. To start with, designers have a significant role to play in ensuring data integrity by grouping attributes to form a high-quality database, avoiding redundant information, update anomalies, and spurious tuples, and reducing the potential null values (Elmasri & Navathe, 2020). Effective ER-to-Relational Database mapping and normalizations can link tables appropriately, save database storage, make semantics easy to interpret, and prevent continuous problems that may arise during database operations. Moreover, keeping the database up-to-date with regular, secure file backups can help preserve data integrity and facilitate system recovery in the event of security breaches or data loss. These measures can effectively mitigate potential

data integrity violations and significantly preserve data consistency within a database. By adopting these strategies, the database can be protected from potential attacks and ensure the data stored in it is trustworthy and reliable.

In addition, to establish full integrity, owner integrity is required as well, indicating that the owner, person or the device that is responsible for the data should be correctly identified (Riley, 2015). There exists some techniques that the attackers used to violate and breach the owner's integrity such as malware, network sniffing, or tools to impersonate others. Those can also be mitigated or prevented by security strategies we discussed in the data confidentiality section, for instance, firewall, system login identities, data encryption with digital signatures.

**Conclusion**

In conclusion, maintaining data integrity and confidentiality is critical for any organization that handles sensitive information, such as SHEIN. To ensure data confidentiality, the company should restrict access to the database and information stored to authorized personnel only and use firewalls, antivirus software, data encryption, and access control to prevent data breaches. Furthermore, triggers, audits, and views can be used to monitor user activities, track access, and limit access to specific data. To maintain data integrity, designers should create high-quality databases, avoid redundant information, update anomalies, and spurious tuples, and reduce null values. Regular, secure backups of the database can also help preserve data integrity and facilitate system recovery in the event of security breaches or data loss. By implementing these practices, SHEIN can maintain and enhance data integrity, ensuring the protection of sensitive information and building trust with customers and clients.

**References**

Bhattacharya, S., Sharma, R.P., & Gupta, A. (2021). Determinants of Consumer Perceptions of the Ethics of Online Retailers: An Investigation Using Confirmatory Factor Analysis. *Vision: The Journal of Business Perspective*.

*Code of business conduct and ethics.* Amazon.com, Inc. - Corporate governance - Documents and charters - Code of Business Conduct and Ethics. (n.d.). Retrieved April 4, 2023, from https://ir.aboutamazon.com/corporate-governance/documents-and-charters/code-of-business-conduct-and-ethics/default.aspx

DESL. (2023, January 9). *Supplier relationship management*. DeSL. Retrieved March 2, 2023, from https://www.desl.net/supplier-vendor-management-compliance-for-fashion-retail-apparel-footwear/#:~:text=Supplier%20Relationship%20Management%20(SRM)%20software,all%20sourcing%20and%20vendor%20management

Elmasri, R., & Navathe, S. (2020). *Fundamentals of Database Systems*. Pearson.

Government of Alberta. (2023). Personal information protection act – overview. Alberta.ca. Retrieved April 4, 2023, from https://www.alberta.ca/personal-information-protection-act-overview.aspx

Government of Canada. (2019, June 21). Government of Canada. Personal Information Protection and Electronic Documents act. Personal Information Protection and Electronic Documents Act. Retrieved April 4, 2023, from https://laws-lois.justice.gc.ca/eng/acts/p-8.6/page-1.html

Government of Canada. (2023, March 13). Government of Canada. Language selection - Innovation, Science and Economic Development Canada Main Site / Sélection de la langue - Site principal d'Innovation, Sciences et Développement économique Canada. Retrieved April 4, 2023, from https://ised-isde.canada.ca/site/innovation-better-canada/en/artificial-intelligence-and-data-act-aida-companion-document

Grewal, H. (2020). A study on Ethical and Social issues in E-commerce. *Strad Research.*

Groover, D. (2021, October 12). *A complete guide to supplier relationship management (SRM)*. JAGGAER. Retrieved March 2, 2023, from https://www.jaggaer.com/blog/complete-guide-srm/#:~:text=Supplier%20relationship%20management%20(SRM)%20is,key%20component%20to%20business%20success

Mckenzie, B. (2022, December 30). Key Data Privacy and Security Laws: Canada: Global Data Privacy & Security Handbook: Baker McKenzie Resource Hub. Home. Retrieved April 3, 2023, from https://resourcehub.bakermckenzie.com/en/resources/data-privacy-security/north-america/canada/topics/key-data-privacy-and-security-laws#:~:text=Generally%2C%20federal%20and%20provincial%20privacy,be%20collected%2C%20used%20or%20disclosed.

Ogunmola, G.A., & Kumar, V. (2020). Web Analytics and Online Retail: Ethical Perspective. *Int. J. Technoethics, 11*, 18-33.

Patel, V. (2023, February 11). *Zara's Supply Chain Management Model*. Ivalua. Retrieved March 2, 2023, from https://www.ivalua.com/blog/supply-chain-management-zara/

Riley, D. D. (2015). *Computational thinking for the modern problem solver*. CRC Press.

SHEIN. (2022). About Us | SHEIN. SHEIN. Retrieved March 2, 2023, from https://ca.shein.com/%20About-Us-a-117.html

**Statement of Individual Contributions**

- Executive Summary - Yiwen Ma
- Context for the Study - Yiwen Ma
- ERD diagram - Keying Zhu
- Tables - Keying Zhu & Yuetong Jiang
- Normalization - Keying Zhu & Yuetong Jiang & Yiwen Ma
- Views - Keying Zhu & Yuetong Jiang & Yiwen Ma
- Queries - Keying Zhu & Yuetong Jiang & Yiwen Ma
- Database Catalog - Yuetong Jiang & Yiwen Ma
- Legal Aspects - Keying Zhu & Yuetong Jiang & Yiwen Ma
- Ethical Aspects - Keying Zhu & Yuetong Jiang & Yiwen Ma
- Security Aspects - Keying Zhu & Yuetong Jiang & Yiwen Ma
- References- Keying Zhu & Yuetong Jiang & Yiwen Ma