

# **INF 1343 Data Modelling and Database Design**

## **Assignment 2: Database Design**

**by**

Kuan Yi Chou(1003850674)

Ruolan Zhang(1004711010)

Yiran Zheng(1003993725)

## **Table of Content**

<b>1. Executive Summary</b>	<b>3</b>
<b>2. Context for the Study</b>	<b>3</b>
<b>3. ER Diagram</b>	<b>5</b>
<b>4. Tables</b>	<b>6</b>
<b>5. Normalization</b>	<b>6</b>
<b>6. Views</b>	<b>8</b>
<b>7. Queries</b>	<b>10</b>
<b>8. Trigger</b>	<b>12</b>
<b>9. Database Catalog</b>	<b>13</b>
<b>10. Legal Aspect</b>	<b>19</b>
<b>11. Ethical Aspect</b>	<b>21</b>
<b>12. Security Aspect</b>	<b>23</b>
<b>Reference</b>	<b>25</b>
<b>Statement of Individual Contributions</b>	<b>25</b>

## 1. Executive Summary

In this report, we examine the structure of the xxx company through different database designs. Through interviews with the boss and an employee and research on a corporate website, our team is able to navigate through the life cycle journey of customers visiting the website. Although we tried to be thorough in our investigation process, we still had to make some assumptions regarding item policy or consumer activities. Our main assumptions are that the boss does not participate in daily work. Each employee's personal information will record which manager recruited them. A customer will only be assigned to one customer service representative, and all of the customer's orders will be handled by the assigned customer service representative. Each customer has their own independent shopping cart. Payment for orders is not transferable.

We demonstrated the flow of the company through three different diagrams in Assignment 1, which were ERD, EERD and the UML diagram. We then compared the strength and weakness of the diagrams and in terms of several real world use cases. Browsing comments from Assignment 1 marker, our group made several adjustments, such as changing entity relations and removing redundant attributes. Finally, we provided everyone with the up-to-date data catalog documenting each of the individual attribute and entity types that were mentioned in the study.

In addition, with in-depth study on the database design and data manipulation, we normalized tables using three different standard normal forms to reduce data redundancy and dependency after inserting manipulated data into each table. After that, we created ten views and three triggers stored in the current dataset, followed by ten more queries extracting data with conditions.

Although we are unable to provide the actual name and information of the company that we included in the study, we hope that the information we gathered can help describe the structure of entities within the company well. We also want to make sure that we were able to describe the relationship between entities clearly. Finally, we want to describe each entity with attributes clearly.

## 2. Context for the Study

The organization that we studied is an online store that sells toys, stationery, homeware, tableware and gifts. It is a one-stop-shop for all things and contains a range of products from different brands. Customers can browse the product in various colors, styles, and price ranges. Choose from different sizes, styles, and designs to suit their individual needs and preferences. The website provides a seamless shopping experience, with easy-to-use search and browse functions, clear product descriptions, and high-quality product images. Customers can add shopping carts and order products through online payment, and have them delivered directly to their doorstep. Customer service (Employee) will update the stock and pack the products upon receipt of the order, and keep track of the status of the customer's order after it has been shipped. When customers need after-sales service or other needs, employees will respond.

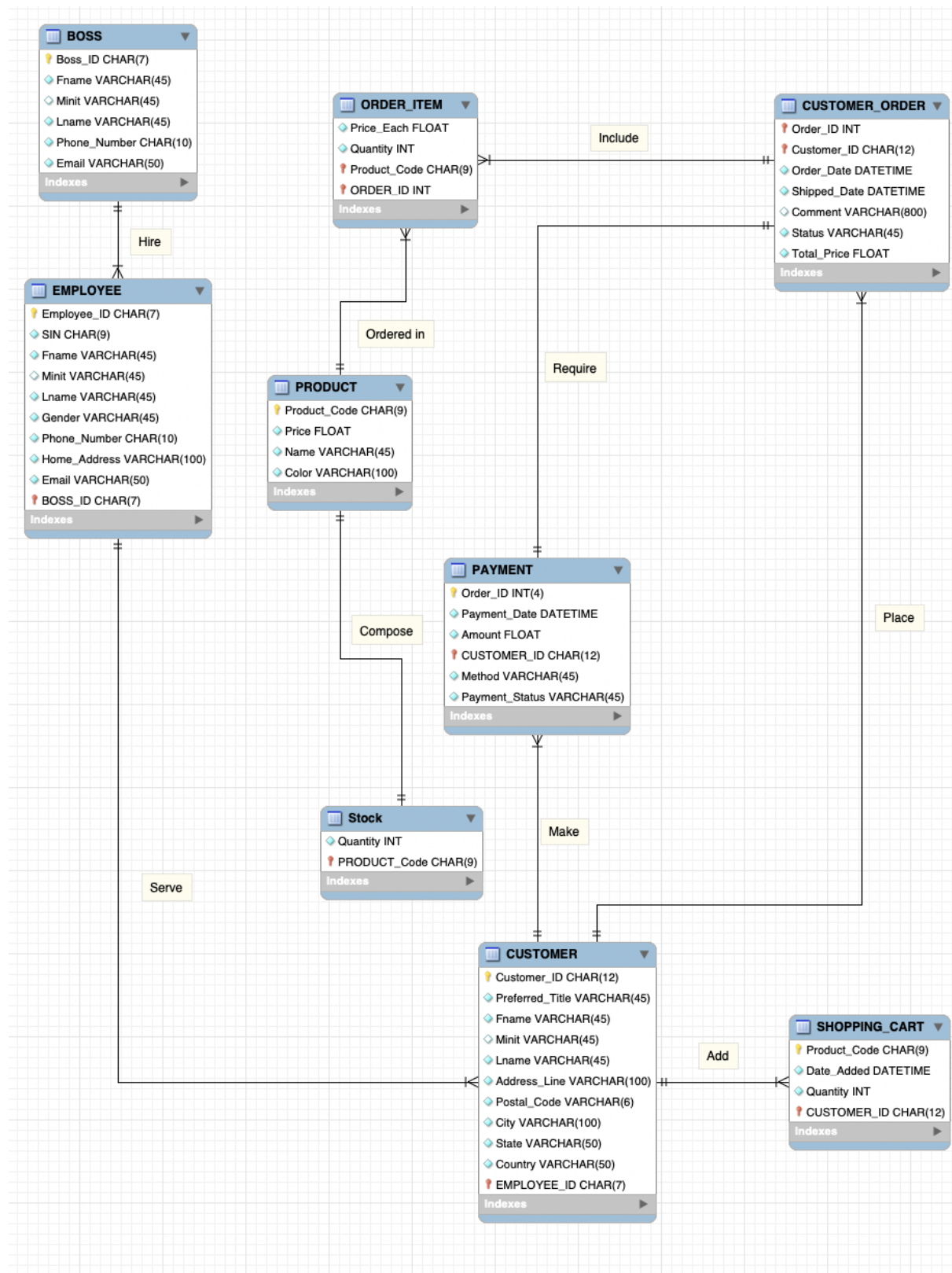
The organization is a small business, which means that there are no specific leading and executive departments aside from the boss. The boss usually handles the human resources department issues. The boss hires employees that handle the day to day transactions, interaction with the customers and the employees will be responsible to handle restock requests as well.

Although we have conducted our interview with the owner, there are still some aspects of which we are not certain. Thus we made assumptions about, the first one is that the boss does not participate at all with customers. We have made another assumption that the definition of customer does not mean that they have to make a purchase, but just need to register their information with the company. Additionally, we assume that individuals who purchase through gift cards would still have to input their information to receive the product. So anyone who creates an account OR purchase through gift cards and such would be a registered customer in our scenario.

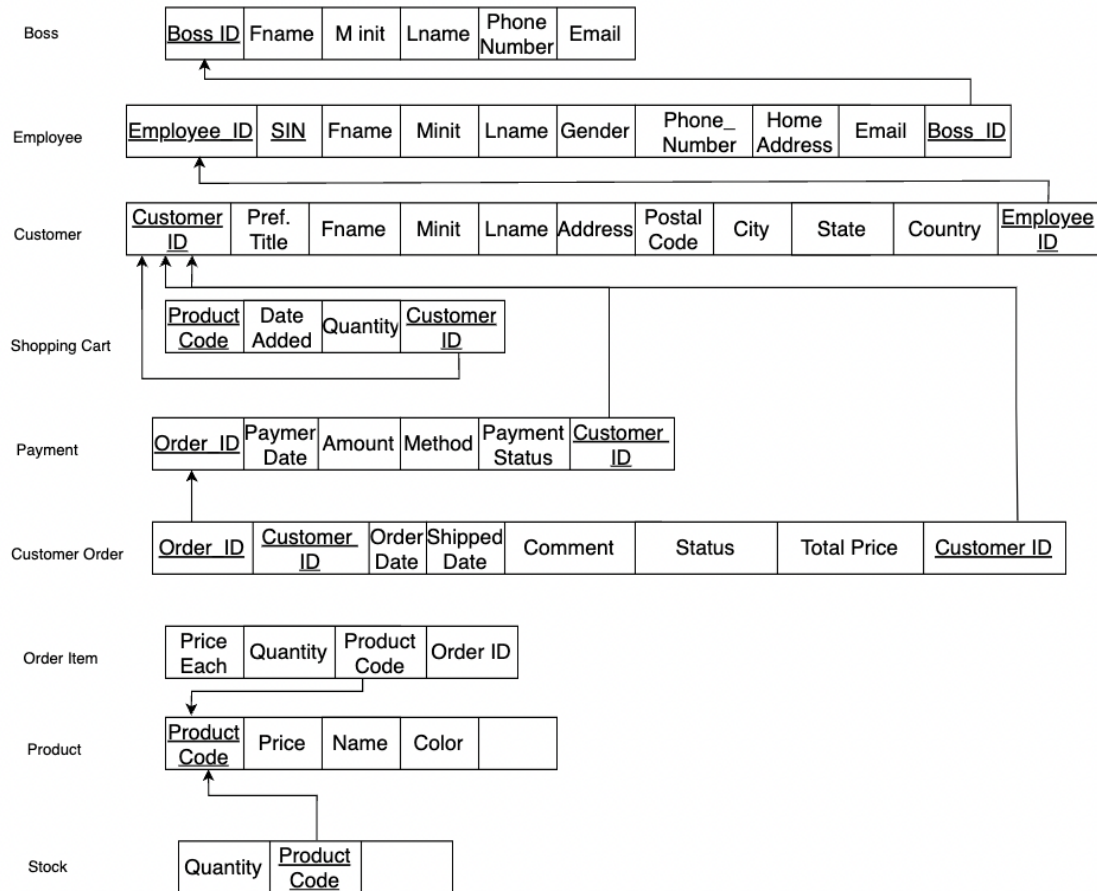
The items sold are divided into some of their unique attributes that we feel like are the best descriptive statistics of the item. For example, stationeries are divided by origination, this is because Japanese and Chinese stationeries are really famous in terms of price, quality and value. Homeware is divided by rooms so that individuals can browse and select the items for the specific rooms that they are interested in. Tableware are divided by occasions, is this a fancy dinner or a normal Thursday night? People need different tableware for different occasions and it will be easier to shop on occasions. Toys are divided by age-groups so that parents and gift givers can find the items to give out more easily. Finally, accessories are divided by gender for both male and female shoppers.

The life cycle of a customer starts by entering the website, where they will be prompted to browse items and load shopping carts. If there are any issues, customers can find employees if needed. After customers finish shopping, they will be prompted to fill in information regarding their identification. After such, they will be prompted to order and pay for the order. Order will then deduct the current stock and compose of the items that the customer has ordered. Individual ordered items included in order will be presented and received from the product region.

### 3. ER Diagram



## 4. Tables



### BOSS Table:

Result Grid						
Filter Rows: <input type="text" value="Search"/>						
Edit:						
Export/Import:						
	<u>Boss_ID</u>	Fname	Minit	Lname	Phone_Number	Email
▶	1000001	Anthony	KY	Chou	4589137201	Anthony.chou@outlook.com
▶	1000002	Summer	RL	Zhang	7126548109	Summer.zhang@outlook.com
▶	1000003	Kiran	YR	Zheng	8315902483	yr9345.zheng@outlook.com
▶	1000004	Samantha	NULL	Nguyen	6041269378	Samantha.nguyen@outlook.com
▶	1000005	Gabrielle	NULL	Rodriguez	2268896054	Gabri.rodriquez@outlook.com
▶	NULL	NULL	NULL	NULL	NULL	NULL

### CUSTOMER Table:

Result Grid

Filter Rows:








Search

Edit:

Export/Import:

	Customer_ID	Preferred_Title	Fname	Minit	Lname	Address_Line	Postal_Code	City	State	Country	EMPLOYEE_ID
▶	AM2023000005	Miss	Ava	NULL	Martinez	130 St. George Street	M4Y0E2	Toronto	Ontario	Canada	2000004
	CK2023000002	Miss	Chloe	JS	Kim	100 John West Way	L4E4E3	Aurora	Ontario	Canada	2000001
	JS2023000003	Mr	Jackson	NULL	Singh	1 Wellington Street	K1A0A3	Ottawa	Ontario	Canada	2000002
	SC2023000001	Mrs	Samantha	YD	Chen	4700 Keele Street	M4L0U8	North York	Ontario	Canada	2000001
	XA2023000004	Mr	Xavier	M	Adams	200 University Avenue West	H1K5E8	Waterloo	Ontario	Canada	2000003
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

### CUSTOMER\_ORDER Table:

Result Grid   Filter Rows: <input type="text" value="Search"/>								Edit:   		Export/Import:  	
Order_ID	Customer_ID	Order_Date	Shipped_Date	Comment	Status	Total_Price					
1478	SC2023000001	2023-01-30 00:00:00	2023-02-02 00:00:00	Make it a gift.	Delivered	61					
1479	CK2023000002	2023-02-02 00:00:00	2023-02-05 00:00:00	NULL	Delivered	23.67					
1480	JS2023000003	2023-02-15 00:00:00	2023-02-16 00:00:00	NULL	Delivered	112.93					
1481	XA2023000004	2023-03-01 00:00:00	2023-03-04 00:00:00	Need Packaging	Delivered	103.91					
1482	AM2023000005	2023-03-12 00:00:00	2023-03-14 00:00:00	NULL	Delivered	56.45					
NULL	NULL	NULL	NULL	NULL	NULL	NULL					

EMPLOYEE Table:

Result Grid

Filter Rows:




Search

Edit:




Export/Import:

Employee_ID	SIN	Fname	Minit	Lname	Gender	Phone_Number	Home_Address	Email	BOSS_ID
2000001	425871639	Olivia	W	Bennett	Female	6843907152	14 York Street, Toronto, ON	Olivia.W.Bennett@outlook.com	1000001
2000002	639857124	Ethan	C	Nguyen	Male	2571469328	10 York Street, Toronto, ON	Ethan.C.Nguyen@outlook.com	1000004
2000003	210495837	Maya	NULL	Patel	Female	9328407616	33 Charles Street, Toronto, ON	Maya.Patel@outlook.com	1000002
2000004	978345216	Alex	B	Rodriguez	Male	7194638251	1080 Bay Street, Toronto, ON	Alex.B.Rodriguez@outlook.com	1000004
2000005	503286914	Lily	NULL	Johnson	Female	5028315746	365 Dundas Street, Toronto, ON	Lily.Johnson@outlook.com	1000005
2000006	721694538	Sebastian	NULL	Lee	Male	8539712860	1002 Yonge Street, Toronto, ON	Sebastian.Lee@outlook.com	1000003
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PRODUCT Table:

Result Grid   Filter Rows: <input type="text" value="Search"/>						Edit: 	
Product_Code	Price	Name	Color				
BP1000001	46.99	Backpack	Yellow				
CC1000001	26.99	Cloud Coffee Cup	Cream with blue spots				
CS1000001	8.99	China Spoon	White				
CS1000002	8.99	China Spoon	Silver				
LP1000001	39.99	Lamp	White				
LP1000002	39.99	Lamp	Black				
MK1000001	2.99	Marker	Green				
MK1000002	2.99	Marker	Blue				
SR1000001	3.99	Sanrio Ruler	Light Blue				
NULL	NULL	NULL	NULL				

SHOPPING\_CART Table:

Result Grid   Filter Rows: <input type="text" value="Search"/>					Edit: 	
Product_Code	Date_Added	Quantity	CUSTOMER_ID			
MK1000002	2023-03-26 00:00:00	5	AM2023000005			
SR1000001	2023-03-10 00:00:00	2	CK2023000002			
BP1000001	2023-03-14 00:00:00	1	JS2023000003			
CC1000001	2023-03-07 00:00:00	1	SC2023000001			
CS1000001	2023-03-20 00:00:00	3	XA2023000004			
NULL	NULL	NULL	NULL			

PAYMENT Table:

Result Grid

Filter Rows:



Search

Edit:




Export/Import:

Order_ID	Payment_Date	Amount	CUSTOMER_ID	Method	Payment_Status	
▶ 1478	2023-01-30 00:00:00	61	SC2023000001	Debit Card	Recieved	
1479	2023-02-02 00:00:00	23.67	CK2023000002	Debit Card	Recieved	
1480	2023-02-15 00:00:00	112.93	JS2023000003	Debit Card	Recieved	
1481	2023-03-01 00:00:00	103.91	XA2023000004	Credit Card	Recieved	
1482	2023-03-12 00:00:00	56.45	AM2023000005	Gift Card	Recieved	
NULL	NULL	NULL	NULL	NULL	NULL	

STOCK Table:

Result Grid   Filter Rows: <input type="text" value="Search"/>		
Quantity	PRODUCT Code	
▶ 19	BP1000001	
38	CC1000001	
50	CS1000001	
50	CS1000002	
20	LP1000001	
18	LP1000002	
67	MK1000001	
105	MK1000002	
63	SR1000001	
NULL	NULL	

ORDER\_ITEM Table:

Result Grid   Filter Rows: <input type="text" value="Search"/>					Edit: 	
Price_Each	Quantity	Product_Code	ORDER_ID			
▶ 46.99	1	BP1000001	1480			
46.99	1	BP1000001	1481			
26.99	2	CC1000001	1478			
26.99	1	CC1000001	1481			
8.99	2	CS1000001	1481			
8.99	1	CS1000002	1479			
39.99	1	LP1000001	1480			
39.99	2	LP1000002	1482			
2.99	2	MK1000001	1479			
2.99	2	MK1000002	1479			
2.99	3	MK1000002	1480			
2.99	2	MK1000002	1482			
3.99	1	SR1000001	1480			
3.99	1	SR1000001	1482			
NULL	NULL	NULL	NULL			



## 5. Normalization

### Identify Primary Keys for each table:

Boss: Boss\_ID(PK)

Customer: Customer\_ID(PK), Employee\_ID(FK)

Customer\_Order: Order\_ID(PK&FK), Customer\_ID(FK)

Employee: Employee\_ID(PK), Boss\_ID(FK)

Order\_Item: (Product\_Code, Order\_ID)(PK)

Payment: Order\_ID(PK), Customer\_ID(FK)

Product: Product\_Code(PK)

Shopping\_Cart: (Product\_Code, Customer\_ID)(PK), Customer\_ID(FK)

Stock: Product\_Code(PK&FK)

1NF violation situation:

1. using row order to convey information
2. Mixing data types within the same column
3. Designing a table without a primary key
4. Strong a repeating group of data items in a single row(multi-values)

Firstly, we are trying to bring tables to the First Normal Form. According to four criterias of 1NF, none of the rules are violated for all tables, so we don't need to split them further.

Moving on to the Second Normal Form, we need to ensure that every non-key attribute in each table is functionally dependent on the entire primary key. We will access each table separately in this section:

1. BOSS table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
2. Customer table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
3. Customer\_Order table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
4. Employee table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
5. Order\_Item table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
6. Payment table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
7. Product table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
8. Shopping\_Cart table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.
9. Stock table:  
Already in 2NF. Every non-key attribute in the table is dependent on the primary key.

Finally, we have to check whether those tables are in 3NF. According to criteria, every non-key attribute in a table should depend on the entire key, and nothing else. Transitive dependencies are not allowed in 3NF.

10. BOSS table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

11. Customer table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

12. Customer\_Order table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

13. Employee table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

14. Order\_Item table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

15. Payment table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

16. Product table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

17. Shopping\_Cart table:

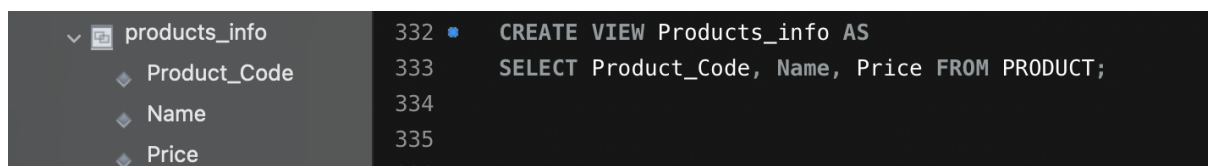
Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

18. Stock table:

Already in 3NF. There is no transitive dependency in this table. Every non-key attribute in a table should depend on the entire key, and nothing else.

## 6. Views

1. Create a view named Products\_info contains all products with their Product\_Code, Fname, Minit, Lname and Price:



```
332 CREATE VIEW Products_info AS
333 SELECT Product_Code, Name, Price FROM PRODUCT;
334
335
336
```

2. Create a view named Payment\_info contains all payments with their Order\_ID, Payment\_Date, Amount, and Method:

<ul style="list-style-type: none"> <li>payment_info <ul style="list-style-type: none"> <li>Order_ID</li> <li>Payment_Date</li> <li>Amount</li> <li>Method</li> </ul> </li> </ul>	<pre> 335 CREATE VIEW Payment_info AS 336     SELECT Order_ID, Payment_Date, Amount, Method FROM PAYMENT; 337 338 339 </pre>
--	--

3. Create a view named Employees\_boss contains all employees with their Fname, Minit, Lnam, Email, and Boss\_ID:

<ul style="list-style-type: none"> <li>employees_boss <ul style="list-style-type: none"> <li>Fname</li> <li>Minit</li> <li>Lname</li> <li>Email</li> <li>Boss_ID</li> </ul> </li> </ul>	<pre> 337 338 CREATE VIEW Employees_boss AS 339     SELECT Fname,Minit,Lname, Email, Boss_ID FROM EMPLOYEE; 340 341 342 343 </pre>
---	--

4. Create a view named Orders\_status contains all orders with their Customer\_ID, Order\_Date, Shipped\_Date, and Status:

<ul style="list-style-type: none"> <li>orders_status <ul style="list-style-type: none"> <li>Customer_ID</li> <li>Order_Date</li> <li>Shipped_Date</li> <li>status</li> </ul> </li> </ul>	<pre> 341 CREATE VIEW Orders_status AS 342     SELECT Customer_ID, Order_Date, Shipped_Date, status FROM CUSTOMER_ORDER; 343 344 345 346 </pre>
--	---

5. Create a view named Shopping\_Carts contains all shopping carts with their Product\_Code, Quantity, and Customer\_ID:

<ul style="list-style-type: none"> <li>shopping_carts_info <ul style="list-style-type: none"> <li>Product_Code</li> <li>Quantity</li> <li>Customer_ID</li> </ul> </li> </ul>	<pre> 344 CREATE VIEW Shopping_Carts_info AS 345     SELECT Product_Code, Quantity, Customer_ID FROM SHOPPING_CART; 346 347 348 </pre>
--	--

6. Create a view named Orders\_total contains all orders with their Customer\_ID, Order\_Date, and Total\_Price:

<ul style="list-style-type: none"> <li>orders_total <ul style="list-style-type: none"> <li>Customer_ID</li> <li>Order_Date</li> <li>Total_Price</li> </ul> </li> </ul>	<pre> 347 CREATE VIEW Orders_total AS 348     SELECT Customer_ID, Order_Date, Total_Price FROM CUSTOMER_ORDER; 349 350 </pre>
--	---

7. Create a view named Products\_stock contains all products with their Product\_Code, Fname, Minit,Lname, Color, and Quantity:

<ul style="list-style-type: none"> <li>products_stock <ul style="list-style-type: none"> <li>Product_Code</li> <li>Name</li> <li>Color</li> <li>Quantity</li> </ul> </li> </ul>	<pre> 350 CREATE VIEW Products_stock AS 351     SELECT PRODUCT.Product_Code, PRODUCT.Name, PRODUCT.Color, STOCK.Quantity 352     FROM PRODUCT 353     INNER JOIN STOCK ON PRODUCT.Product_Code = STOCK.Product_Code; 354 355 </pre>
---	---

8. Create a view named Customer\_locations contains all customers with their Fname, Minit, Lname, Address\_line, City, State, Postal\_Code, and Country:

Views	354
customers_location	355
Customer_ID	356
Fname	357
Minit	358
Lname	359
Address_line	360
City	361
State	362
Postal_Code	363
Country	364
	365

```

354 CREATE VIEW Customers_location AS
355 SELECT Customer_ID, Fname, Minit, Lname, Address_line, City, State, Postal_Code, Country FROM CUSTOMER;
356
357
358
359
360
361
362
363
364
365

```

9. Create a view named Employees\_contact contains all employees with their SIN, Fname, Minit, Lname, and Phone\_Number:

employees_contact	358
SIN	359
Fname	360
Minit	361
Lname	362
Phone_Number	363
	364

```

358 CREATE VIEW Employees_contact AS
359 SELECT SIN, Fname, Minit, Lname, Phone_Number FROM EMPLOYEE;
360
361
362
363
364

```

10. Create a view named Order\_items\_info contains all order items with their Price, Quantity, and Product\_Code:

order_items_info	361
Price_Each	362
Quantity	363
Product_Code	364
	365

```

361 CREATE VIEW Order_items_info AS
362 SELECT Price_Each, Quantity, Product_Code FROM ORDER_ITEM;
363
364
365

```

## 7. Queries

1. Select all customers who live in the city of "North York":

```

1 SELECT *
2 FROM INF1343_A2.CUSTOMER
3 WHERE city = 'North York';

```

Customer_ID	Preferred_Title	Fname	Minit	Lname	Address_Line	Postal_Code	City	State	Country	EMPLOYEE_ID
SC2023000001	Mrs	Samantha	YD	Chen	4700 Keele Street	M4L0U8	North York	Ontario	Canada	2000001

2. Select all customers who have a preferred title of "Miss." or "Mrs.":

```

1 SELECT *
2 FROM INF1343_A2.CUSTOMER
3 WHERE preferred_title IN ('Miss', 'Mrs');

```

Customer_ID	Preferred_Title	Fname	Minit	Lname	Address_Line	Postal_Code	City	State	Country	EMPLOYEE_ID
AM2023000005	Miss	Ava	NULL	Martinez	130 St. George Street	M4Y0E2	Toronto	Ontario	Canada	2000004
CK2023000002	Miss	Chloe	JS	Kim	100 John West Way	L4E4E3	Aurora	Ontario	Canada	2000001
SC2023000001	Mrs	Samantha	YD	Chen	4700 Keele Street	M4L0U8	North York	Ontario	Canada	2000001

3. Select all orders with a total price greater than \$100:

```

1 • SELECT *
2 FROM INF1343_A2.CUSTOMER_ORDER
3 WHERE total_price > 100;

```

	Order_ID	Customer_ID	Order_Date	Shipped_Date	Comment	Status	Total_Price
▶	1480	JS2023000003	2023-02-15 00:00:00	2023-02-16 00:00:00	NULL	Delivered	112.93
	1481	XA2023000004	2023-03-01 00:00:00	2023-03-04 00:00:00	Need Packaging	Delivered	103.91

4. Select the Customer\_ID, name and Preferred\_Title of all customers who have a postal code starting with "M4":

```

1 • SELECT Customer_ID, Fname, Minit, Lname, Preferred_Title
2 FROM INF1343_A2.CUSTOMER
3 WHERE postal_code LIKE 'M4%';

```

	Customer_ID	Fname	Minit	Lname	Preferred_Title
▶	AM2023000005	Ava	NULL	Martinez	Miss
	SC2023000001	Samantha	YD	Chen	Mrs

5. Select the Boss\_ID who hired more than one employees:

```

1 • SELECT Boss_ID, COUNT(*) AS Count_hire
2 FROM INF1343_A2.EMPLOYEE
3 GROUP BY Boss_ID
4 HAVING Count_hire > 1;

```

	Boss_ID	Count_hire
▶	1000004	2

6. Select the name and email of all boss whose last name Start with the string "Zh":

```

1 • SELECT Fname, Minit, Lname, Email
2 FROM INF1343_A2.Boss
3 WHERE Lname LIKE 'Zh%';

```

	Fname	Minit	Lname	Email
▶	Summer	RL	Zhang	Summer.zhang@outlook.com
	Kiran	YR	Zheng	yr9345.zheng@outlook.com

7. Select employees's email who has a middle name:

```

1 • SELECT *
2 FROM INF1343_A2.EMPLOYEE
3 WHERE Minit IS NOT NULL;

```

Employee_ID	SIN	Fname	Minit	Lname	Gender	Phone_Number	Home_Address	Email	BOSS_ID
▶ 2000001	425871639	Olivia	W	Bennett	Female	6843907152	14 York Street, Toronto, ON	Olivia.W.Bennett@outlook.com	1000001
2000002	639857124	Ethan	C	Nguyen	Male	2571469328	10 York Street, Toronto, ON	Ethan.C.Nguyen@outlook.com	1000004
2000004	978345216	Alex	B	Rodriguez	Male	7194638251	1080 Bay Street, Toronto, ON	Alex.B.Rodriguez@outlook.com	1000004

8. Select all customers who have made a payment using the "credit card" method:

```

1 • SELECT *
2 FROM INF1343_A2.CUSTOMER
3 WHERE Customer_ID IN (
4     SELECT customer_ID
5     FROM INF1343_A2.PAYMENT
6     WHERE method = 'credit card'
7 );

```

Customer_ID	Preferred_Title	Fname	Minit	Lname	Address_Line	Postal_Code	City	State	Country	EMPLOYEE_ID
▶ XA2023000004	Mr	Xavier	M	Adams	200 University Avenue West	H1K5E8	Waterloo	Ontario	Canada	2000003

9. Select all orders along with their respective customer information

```

1 • SELECT o.Order_ID, o.Order_Date, o.Total_Price, c.Fname, c.Minit, c.Lname, c.Address_Line, c.city, c.state, c.country
2 FROM INF1343_A2.CUSTOMER_ORDER o
3 JOIN INF1343_A2.CUSTOMER c ON o.Customer_ID = c.Customer_ID;

```

Order_ID	Order_Date	Total_Price	Fname	Minit	Lname	Address_Line	city	state	country
▶ 1478	2023-01-30 00:00:00	61	Samantha	YD	Chen	4700 Keele Street	North York	Ontario	Canada
1479	2023-02-02 00:00:00	23.67	Chloe	JS	Kim	100 John West Way	Aurora	Ontario	Canada
1480	2023-02-15 00:00:00	112.93	Jackson	NULL	Singh	1 Wellington Street	Ottawa	Ontario	Canada
1481	2023-03-01 00:00:00	103.91	Xavier	M	Adams	200 University Avenue West	Waterloo	Ontario	Canada
1482	2023-03-12 00:00:00	56.45	Ava	NULL	Martinez	130 St. George Street	Toronto	Ontario	Canada

10. Select all products along with their respective stock

```

1 • SELECT p.Product_Code, p.Name, p.Color, s.Quantity
2 FROM INF1343_A2.PRODUCT p
3 JOIN INF1343_A2.Stock s ON p.Product_code = s.PRODUCT_Code;

```

Product_Code	Name	Color	Quantity
▶ BP1000001	Backpack	Yellow	19
CC1000001	Cloud Coffee Cup	Cream with blue spots	38
CS1000001	China Spoon	White	50
CS1000002	China Spoon	Silver	50
LP1000001	Lamp	White	20
LP1000002	Lamp	Black	18
MK1000001	Marker	Green	67
MK1000002	Marker	Blue	105
SR1000001	Sanrio Ruler	Light Blue	63



## 8. Trigger

1. A trigger that can automatically update the stock of products after a new order is placed by customers. It decreases the stock quantity of every product.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' panel is open, showing a tree view of database objects. The 'Triggers' folder is expanded, and the trigger 'update\_stock\_on\_order' is selected. On the right, the SQL editor displays the following code:

```

1 DELIMITER $$
2 CREATE TRIGGER update_stock_on_order AFTER INSERT
3 ON ORDER_ITEM
4 FOR EACH ROW
5 BEGIN
6     UPDATE Stock SET Quantity = Quantity - ORDER_ITEM.Quantity
7     WHERE Product_Code = ORDER_ITEM.Product_Code;
8 END$$
9 DELIMITER ;

```

2. Trigger for Status. If there is an order date and no shipped date, then the status will be ordered. If there is an updated shipped date, then the status will be shipped.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' panel is open, showing a tree view of database objects. The 'Triggers' folder is expanded, and the trigger 'update\_order\_status' is selected. On the right, the SQL editor displays the following code:

```

1 DELIMITER $$
2
3 CREATE TRIGGER update_order_status BEFORE UPDATE
4 ON CUSTOMER_ORDER
5 FOR EACH ROW
6 BEGIN
7     IF NEW.Shipped_Date IS NOT NULL THEN
8         SET NEW.Status = 'Shipped';
9     END IF;
10 END$$
11
12 DELIMITER ;

```

3. A trigger to record the payment date in the Payment table. When the payment status shows 'Received', the system will record the current time as payment time.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' panel is open, showing a tree view of database objects. The 'Triggers' folder is expanded, and the trigger 'record\_payment\_date' is selected. On the right, the SQL editor displays the following code:

```

1 DELIMITER $$
2
3 CREATE TRIGGER record_payment_date BEFORE UPDATE
4 ON PAYMENT
5 FOR EACH ROW
6 BEGIN
7     IF NEW.Payment_Status = 'Received' THEN
8         SET NEW.Payment_Date = NOW();
9     END IF;
10 END$$
11
12 DELIMITER ;

```

## 9. Database Catalog

Attributes	Entity	Type	Domain	Constrain	Key
Product_Code	Product	STRING	CHAR(9) 9 character combination of letters and	<ul style="list-style-type: none"> <li>- Not NULL</li> <li>- Unique</li> </ul>	Primary Key

			number; first two characters are letters		
Price		FLOAT	The set of floats larger than 0.00	<ul style="list-style-type: none"> <li>- Not NULL</li> <li>- Currency: Canadian dollars (CAD)</li> </ul>	
Name		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	
Color		STRING	VARCHAR(100) Maximum of 20 characters combination (eg. Red spots with yellow lines)	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	
Order_ID	<b>Order Item</b>	INTEGER	INT(4) (eg. 1401)	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	Foreign key
Price_Each		FLOAT	The set of floats larger than 0.00	<ul style="list-style-type: none"> <li>- Not NULL</li> <li>- Currency: Canadian dollars (CAD)</li> </ul>	
Quantity		INTEGER	Integers larger than 0	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	
Product_Code		STRING	CHAR(9) 9 character combination of letters and number	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	Foreign key
Order_ID	<b>Customer_Order</b>	INTEGER	INT(4) (eg. 1401)	<ul style="list-style-type: none"> <li>- Not NULL</li> <li>- Unique</li> </ul>	Primary key Foreign key
Customer_ID		STRING	CHAR(12) 12 character combination of letters and number; first two characters are letters	<ul style="list-style-type: none"> <li>- Not NULL</li> </ul>	Foreign key



Order_Date		DATE	MM-DD-YYYY (eg. 03-06-2023)	- Not NULL	
Shipped_Date		DATE	MM-DD-YYYY (eg. 03-06-2023)	- Not NULL	
Comment		STRING	VARCHAR(800) (eg. please handle the package to my concierge)		
Status		ENUMERATED	ENUM (‘Confirmed’, ‘Processing’, ‘Shipping’, ‘Delivered’, ‘Canceled’)	- Not NULL	
Total_Price		FLOAT	The set of floats larger than 0.00	- Not NULL - Currency: Canadian dollars (CAD)	
Product_Code	<b>Stock</b>	STRING	CHAR(9) 9 character combination of letters and number	- Not NULL	Foreign key
Quantity		INTEGER	Integers larger than 0	- Not NULL	
Employee_ID	<b>Employee</b>	STRING	CHAR(7) A series of 7 integers	- Not NULL - Unique	Primary key
SIN		STRING	CHAR(9) A series of 9 integers	- Not NULL - Unique	
Fname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	
Minit		STRING	VARCHAR(45) Maximum of 45 alphabetical		

			letters combination		
Lname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	
Gender		ENUMERATED	ENUM('Male', 'Female', 'Prefer not to say')	- Not NULL	
Phone_Number		STRING	CHAR(10) A series of 10 integers	- Not NULL	
Home_Address		STRING	VARCHAR(100) Maximum of 100 alphabetical letters and numbers combination	- Not NULL	
Email		STRING	VARCHAR(50) A series of characters with length smaller than 50 letters	- Not NULL	
BOSS_ID		STRING	CHAR(7) A series of 7 integers	- Not NULL	Foreign key
Customer_ID	<b>Customer</b>	STRING	CHAR(12) 12 characters combination of letters and numbers; first two characters are letters	- Not NULL - Unique	Primary key
Preferred_Title		ENUMERATED	ENUM('Mr', 'Mrs', 'Miss', 'Ms')	- Not NULL	
Fname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	

Minit		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination		
Lname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	
Address_Line		STRING	VARCHAR(100) Maximum of 100 alphabetical letters and numbers combination	- Not NULL	
Postal_Code		STRING	CHAR(6) A combination of letters and numbers	- Not NULL	
City		STRING	VARCHAR(100) Maximum of 100 alphabetical letters combination	- Not NULL	
State		STRING	VARCHAR(50) Maximum of 50 alphabetical letters combination	- Not NULL	
Country		STRING	VARCHAR(50) Maximum of 50 alphabetical letters combination	- Not NULL	
Employee_ID		STRING	CHAR(7) A series of 7 integers	- Not NULL	Foreign key
Product_Code	<b>Shopping Cart</b>	STRING	CHAR(9) 9 character combination of letters and number	- Not NULL - Unique	

Date_Added		DATE	MM-DD-YYYY (eg. 03-06-2023)	- Not NULL	
Quantity		INTEGER	Integers larger than 0	- Not NULL	
CUSTOMER_ID		STRING	CHAR(12) 12 characters combination of letters and numbers; first two characters are letters	- Not NULL	Foreign key
Order_ID	<b>Payment</b>	INTEGER	INT(4) (eg. 1401)	- Not NULL - Unique	Primary key
Payment_Date		DATETIME	MM-DD-YYYY (eg. 03-06-2023)	- Not NULL	
Amount		FLOAT	Float larger than 0.00	- Not NULL	
Method		ENUMERATED	ENUM('Debit Card', 'Credit Card', 'Gift Card')	- Not NULL	
Payment_Status		ENUMERATED	ENUM('Received', 'Declined', 'In progress')	- Not NULL	
CUSTOMER_ID		STRING	CHAR(12) 12 characters combination of letters and numbers; first two characters are letters	- Not NULL	Foreign key
Boss_ID	<b>Boss</b>	STRING	CHAR(7) A series of 7 integers	- Not NULL - Unique	Primary key
Fname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	

Minit		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination		
Lname		STRING	VARCHAR(45) Maximum of 45 alphabetical letters combination	- Not NULL	
Phone Number		STRING	CHAR(10) A series of 10 integers	- Not NULL	
Email		STRING	VARCHAR(50) A series of characters with length smaller than 50 letters	- Not NULL	

## 10. Legal Aspect

Legal aspects: Briefly discuss existing legal aspects protecting the data within the scope of your work. Consider the Canadian law (each student individually, followed by a group discussion). VIEW IN GROUP

Legal Aspects (2 pages, 0.5 pages per student + 0.5 pages group conclusion)

### Kuan Yi Chou

In our observation, the data protection of the firm needs to be improved. According to PIPEDA, institutions need to be able to provide services to customers on the bare minimum amount of information required. In our overall settings, we do have some information that is not necessary but is required. Since our business does not have any security, anti-money laundering or terrorism attached to it, we do not have to run background checks upon the users. Thus we do not necessarily have to have the Name of each individual who is ordering the utensils. It can be changed to not "not null".

Aside from that, we are able to track the corresponding employee to customer and order to deliver well so that if there are any disputes in the order there will be information available that we can backtrack on.

In my opinion, the legal aspect of the overall data collection process is done quite well as Customer ID is being referred to as a foreign key and not their private information in most occasions, even though there may exist a need to remove some inputs, it may be purely a philosophical question of whether name is crucial in this sense.

### Ruolan Zhang

Talking about the legal aspect, Canada has proposed a set of federal and provincial statutes aiming to perform data protection for each individual project. The most famous federal one is

“The Personal Information Protection and Electronic Documents Act”, abbreviated as PIPEDA. This act was revised in May 2019, setting up a few requirements for organizations who would like to collect personal or business information. It requires obtaining consent before performing data collection and these collected data can only be used for the purpose of collecting. Although different provinces in Canada have their own provincial privacy policy laws that are similar to PIPEDA, all business activities or academic researches operated are subject to PIPEDA regardless of their provinces.

In our project, we collected data about the information system of one stationary shop in downtown Toronto by interviewing its owner. We clearly claimed that all information gathered will only be used for the purpose of completing Assignment from the course INF1343 at the University of Toronto. We outlined our intent of study and clarified mutual expectations. After obtaining the consent from the owner, our group started the interview with this person. The name of the stationary shop is concealed and data entries for each table are fabricated. All collected confidential information is excluded from discussion and written reports.

### **Yiran Zheng**

The legal aspects of data protection and privacy are essential for handling personal data. There are several Canadian laws and regulations that protect the privacy and security of personal data, such as the Personal Information Protection and Electronic Documents Act (PIPEDA), which applies to private sector organizations that collect, use or disclose personal information in the course of commercial activities, the Privacy Act applies to federal government institutions and regulates how personal information is collected, used, and disclosed by these institutions. These laws and regulations set out rules for obtaining consent, limiting collection, use and disclosure of personal information, as well as safeguarding the information through appropriate security measures.

When we set up this database for the small business organization selected for our assignment, the collection and storage of personal data about our customers, such as names, addresses, telephone numbers and email addresses, were subject to Canadian laws and regulations that protect the privacy and security of personal data. If we put ourselves in the shoes of those who provide information, we wouldn't want our privacy to be exposed for any purpose, so we handle data very carefully. We therefore obtained the owner's consent before collecting and using the data from this commercial organization, and we informed the owner of what data was being collected, how it would be used, and with whom it would be shared. The data we collect is placed into a database we have created and no information is disclosed to outside parties, the database is only used for research in our project operations, and we do not collect and use real customer information from the business in our operations. In addition to following the legal aspects of data protection and privacy, building the robustness of data modeling and database design is helpful for archive legal obligations and protect the privacy and security of personal data.

In conclusion, our group has been diligent in ensuring the data protection and privacy aspects of the information system for the stationary shop in downtown Toronto. As we have considered various Canadian laws and regulations, such as PIPEDA and the Privacy Act, we have taken the necessary steps to comply with these requirements while setting up the database for the selected small business organization. The legal aspects of data collection and protection have been well addressed in our project, with careful attention given to

obtaining consent, limiting collection, and safeguarding the information through appropriate security measures.

### **Group Conclusion**

Throughout our study, we have been transparent with the owner regarding the collection and need of the information. Moreover, we have assured that all data that is confidential and privacy related have been masked or are fabricated data.

Our database models the lifespan of the delivery of the items well. However, we recognize that there is room for improvement in the data protection of our company. We suggest improving the data collection process to exclude some unnecessary inputs, like the name of each individual ordering utensils, which is not crucial for our project. By refining the data collection process, it will help us to lessen the load of our legal work. Our group has managed to balance the need for accurate and comprehensive data while protecting the privacy of individual users.

Ultimately, our group's focus on the legal aspects of the company's database structure is centered around data protection and privacy demonstrated by abiding to Canadian laws and regulations. We understand that safeguarding personal data is one of the most important aspects of privacy. By remaining vigilant in our data handling practices and continuously improving our database design, we strive to protect the customer and enhance their shopping experience.

## **11. Ethical Aspect**

Ethical aspects: Search and discuss best ethical practices for dealing with data (each student individually, followed by a group discussion)

Ethical Aspects (2 pages, 0.5 pages per student + 0.5 pages group conclusion)

### **Kuan Yi Chou**

In our design, we have an utmost standard in respecting the Ethical Aspect of Data that we collect, store and use. Privacy is our main concern, so individuals do not have to worry about their information being used outside of the storage system that stores it. Everything that needs to call customer identification will need to be referred to through the Customer\_ID and tracked in that sense, allowing more security in the way. We also respect social equality and fairness as our company does not discriminate by gender, race or any other aspects that may trigger individuals. Our agents will be held accountable for their actions as they are being linked to the customer they serve. If there are any issues with an order, we can get the help to the individual customer who needs support right away.

However, we may not be as transparent with our data usage as possible. Data collected would be used in the backend to provide recommendations and sorting algorithms that will help promote other related products. This may not be known to a regular customer, however it should be mentioned in the fineprints of our terms and conditions. Which usually nobody checks on.

### **Ruolan Zhang**

Data ethics specifically stands for the moral duty of gathering, protecting and using personal information. In this aspect, people are ruled by morality and self-regulation instead of proposed acts or legal documentations. As data scientists, we have the responsibility to follow data ethics and moral principles while handling data-related research. Maybe currently we are not people who manipulate data or design databases for business activities, understanding data ethics is still pretty crucial since we will have the ability to discern any instance that violates the moral duty of dealing with data. In this way, we are able to better protect confidential information provided by different organizations.

There are five major principles of data ethics considering taking part into business activities or academic research: ownership, transparency, privacy, intention and outcomes. Besides the ownership of data for individuals, the owner of the data must have rights to know how data scientists plan to collect data and how those collected data will be implemented in further research according to data transparency. Our group detailedly talked about our project plans and requirements with the owner of the stationary shop before collecting data, so this project is totally transparent from the perspective of the owner. Keeping data as privacy is also ensured as all data entities are completely manipulated by our group members. In addition, our intention of collecting data is to help the stationary shop upgrade its information system and optimize efficiency. Since then, our project is still half-way finished so we are not able to comprehensively assess the outcome of data analysis, but we will try our best to prevent any potential damage that may be imposed on the source of the data.

### **Yiran Zheng**

When building this database, we need to first determine the content of the data we need, and during the collection process, we need to ensure that the data collected is accurate, up-to-date, and relevant to the purpose for which it was collected. We avoid collecting unnecessary data. This helps minimize the risk of data breaches and protects the privacy of individuals. The content of our entire database is related to the store, so the content of our data revolves around customers and products, such as basic customer information, product names of the store, and web orders. We only enter and anonymize data that is relevant to the purpose of the job, we do not collect or enter data that is not relevant to the job. We collect, use and build the database transparently and with the consent of the owner, who is informed that the data we collect is for course research purposes only and is not intended for any commercial use. Our compliance with ethical standards ensures that we collect, store, use, and access data in an ethical manner, which further ensures that our database is ethical, responsible, and respectful of the rights and privacy of individuals.

Without violating any ethics, since the store relies primarily on buying and selling merchandise for profit, we hope that data modeling will help the store analyze sales data to identify patterns and trends, such as which products are most popular at certain times of the year. This could help the store make informed decisions about which products to stock and how many to order. In addition, the database would allow stores to track inventory levels, sales trends and order details, helping them better serve their customers. However, our database has not been put into real use, so we will not be using it for any business practices. The above mentioned are the benefits we expect the database to bring to the stores once it is put into use.

### **Group Discussion**



In conclusion, our group is very diligent in the ethical aspects of data collection, storage, and usage in the design for our database. We have very high privacy standards and look upon social equality, ensuring that the information of individuals is treated with respect and secured within our storage system. By using the Customer\_ID to track customer identification and linking agents to the customers they serve, we have established a secure and accountable framework for handling customer data.

Our commitment to data ethics is demonstrated by our five major principles: ownership, transparency, privacy, intention, and outcomes. We will maintain our transparency to all stakeholders involved with the shop. However, we recognize that there is room for improvement in our transparency regarding data usage. While we use the collected data in the backend for recommendations and sorting algorithms, this may not be apparent to the average customer. In the future, we plan on releasing notes on how individuals' data have been used so that customers can track the usage.

As we continue to develop and refine our database, we remain committed to upholding ethical standards in our data collection, storage, and usage. We want to ensure that we maintain the trust of the individuals whose information we handle and demonstrate our dedication to responsible and respectful data practices.

## **12. Security Aspect**

Security aspects: Search and discuss best security practices for dealing with data (each student individually, followed by a group discussion).

Security Aspects (2 pages, 0.5 pages per student + 0.5 pages group conclusion)

### **Kuan Yi Chou**

Security is an important aspect of any database. We think that we have done a decent structure in our miniworld as we have isolated different parts of the company so that one breached department will not affect the other. We would also make it difficult for attackers to gain entrance to our main databases as the owner/boss, employee and customer information and identity will be stored separately. There is no reason to keep all customer files in one location and we may be able to develop a plan to store them more securely. Our communication will be guarded by a firewall. Information will be defended deeply as well.

In the future, it will be a great idea to regularly implement backup files to make sure that we have another copy of data available if the original is corrupted. It will also be a good idea to encrypt data with a decrypt key so that even if any message and data is stolen it will not be as easy to access the information that is stored within because of the encryption method protecting it.

### **Ruolan Zhang**

Preventing data from being threatened by any internal or external danger is another essential practice that data scientists should learn. Every organization deals with large amounts of data every business day so they must adopt different cybersecurity strategies to make their information become more cyber resilient. The baseline is, no matter which

devices or technologies store the required data, data security practice is mandatorily implemented, otherwise organizations will be fined accordingly.

There are several methods that could help with data security: data encryption, data masking and data resilience. Data encryption encodes information with algorithms, so the only way to decrypt is to provide encryption keys. Two of the famous practices include one-way authentication and two-way authentication. Data masking is much easier to understand, which is to mask specific areas of the database to prevent potential hack intrusion. For example, when we login to a website, the input password is always masked with special signs. In addition, data resilience is a practice to prevent the loss of data due to emergencies by making a copy in advance. However, those methods can only help with preventing external threats but not internal threats. To minimize the possibility of insider threats, organizations should better manage employees who carry sensitive information or have special access to core data.

### **Yiran Zheng**

In today's digital age, where most information is stored electronically, data security is a critical issue for any database that collects and stores information. The information used in this assignment was collected and used with the authorization and consent of the individuals involved, and we anonymized and obfuscated all collected information and did not share our database with anyone outside the group assignment.

If the data designed in our assignment is put to real use, it is necessary for the designer to implement strong access control in the first place. Our database stores personal data of customers, employees, and bosses, so we should use encryption to protect the data in the database. Access to the data should be protected by passwords and restricted to authorized individuals only to prevent unauthorized access or tampering. Any access to the database should be logged and monitored. Secondly, it is important to also back up the data. Regularly backing up the database and storing the backups in a secure location ensures that the data can be recovered in the event of a disaster or security breach. Finally, we also need to design a process for identifying security breaches, controlling them, and notifying affected parties. Ensure that any security breaches are identified in the database at the earliest possible stage and fixed quickly and securely. For users, they should not authorize access or tell someone their password, as this increases the risk of data leakage and information security issues. If many people have access to data or know passwords, it is difficult to control who can access and modify data, and to track and identify unauthorized access. In addition, if data is accessed accidentally or maliciously, this could lead to user privacy breaches, financial losses, reputational damage, and other issues that would have irreversible effects on both the commercial organization and the individual. For stores, the organization needs to protect data, implement multi-factor authentication for user logins, conduct regular security training for employees, build trust with customers, and prevent potential financial and reputational damage from data breaches.

### **Group Conclusion**

In conclusion, security is the biggest concern for any database, especially in regards to sensitive and personal information. While designing the database, our group has been making sure that the data we collect, store, and use is secure. The method that we used is called defending deeply, by isolating different parts of the company and storing owner,

employee, and customer information separately, we have created a more secure environment for our data.

Aside from defending deeply, methods such as data encryption, data masking, and data resilience are all available to make our database more secure. These techniques can help protect the data from external threats. Implementing strong access control, password protection, and employee management can minimize the possibility of inner threats. Furthermore, it is crucial to have a process in place for identifying and addressing security breaches promptly and effectively.

As we move forward, it is essential that we continue to prioritize data security. Methods that can help data security include regularly backing up our data, encryption methods, and strong access control. By monitoring the database, we can identify unauthorized access and protect the security of the data.

In addition, we should consider implementing multi-factor authentication for user logins and conduct training for employees. This will help build trust with customers and prevent potential financial damage that may result from data breaches.

Ultimately, our commitment to data security is to make sure that all of our stakeholders' information is safe and to make them believe that we are a sound company.

## **Reference**

Due to the need for privacy, we are unable to provide the citation of the online store that we have interviewed.

## **Statement of Individual Contributions**

Executive Summary: edit from Assignment 1

Context for the Study: edit from Assignment 1

ERD: Kuan Yi Chou, Ruolan Zhang, YiRan Zheng

Tables: Kuan Yi Chou

Normalization: Kuan Yi Chou, Ruolan Zhang, YiRan Zheng

View: Kuan Yi Chou, Ruolan Zhang, YiRan Zheng

Query: Kuan Yi Chou, Ruolan Zhang, YiRan Zheng

Triggers: Kuan Yi Chou, Ruolan Zhang, YiRan Zheng

Data catalog: Kuan Yi Chou & Ruolan Zhang & YiRan Zheng

Legal aspect, Ethical aspect, Security Aspect: :Kuan Yi Chou & Ruolan Zhang & YiRan Zheng

Ideas were generated between all three individuals.

## Appendix

### Create Database & Data Insert:

-- MySQL Script generated by MySQL Workbench

-- Sat Mar 25 03:57:54 2023

-- Model: New Model    Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS,  
FOREIGN\_KEY\_CHECKS=0;

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';

-- -----

-- Schema mydb

-- -----

-- -----

-- Schema INF1343\_A2

-- -----

-- -----

-- Schema INF1343\_A2

-- -----

CREATE SCHEMA IF NOT EXISTS `INF1343\_A2` DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4\_0900\_ai\_ci ;

USE `INF1343\_A2` ;

-- -----

-- Table `INF1343\_A2`.`BOSS`

```
-----  
  
CREATE TABLE IF NOT EXISTS `INF1343_A2`.`BOSS` (  
  `Boss_ID` CHAR(7) NOT NULL,  
  `Fname` VARCHAR(45) NOT NULL,  
  `Minit` VARCHAR(45) NULL DEFAULT NULL,  
  `Lname` VARCHAR(45) NOT NULL,  
  `Phone_Number` CHAR(10) NOT NULL,  
  `Email` VARCHAR(50) NOT NULL,  
  PRIMARY KEY (`Boss_ID`),  
  UNIQUE INDEX `ID_UNIQUE` (`Boss_ID` ASC) VISIBLE)  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb4  
  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `INF1343_A2`.`EMPLOYEE`  
-----
```

```
CREATE TABLE IF NOT EXISTS `INF1343_A2`.`EMPLOYEE` (  
  `Employee_ID` CHAR(7) NOT NULL,  
  `SIN` CHAR(9) NOT NULL,  
  `Fname` VARCHAR(45) NOT NULL,  
  `Minit` VARCHAR(45) NULL DEFAULT NULL,  
  `Lname` VARCHAR(45) NOT NULL,  
  `Gender` VARCHAR(45) NOT NULL,  
  `Phone_Number` CHAR(10) NOT NULL,  
  `Home_Address` VARCHAR(100) NOT NULL,  
  `Email` VARCHAR(50) NOT NULL,
```

```
`BOSS_ID` CHAR(7) NOT NULL,  
  
PRIMARY KEY (`Employee_ID`, `BOSS_ID`),  
  
UNIQUE INDEX `ID_UNIQUE` (`Employee_ID` ASC) VISIBLE,  
  
UNIQUE INDEX `SIN_UNIQUE` (`SIN` ASC) VISIBLE,  
  
INDEX `fk_EMPLOYEE_BOSS1_idx` (`BOSS_ID` ASC) VISIBLE,  
  
CONSTRAINT `fk_EMPLOYEE_BOSS1`  
  
FOREIGN KEY (`BOSS_ID`)  
  
REFERENCES `INF1343_A2`.`BOSS` (`Boss_ID`)  
  
ON DELETE CASCADE  
  
ON UPDATE CASCADE)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb4  
  
COLLATE = utf8mb4_0900_ai_ci;
```

```
--  
-----  
-- Table `INF1343_A2`.`CUSTOMER`  
-----  
--
```

```
CREATE TABLE IF NOT EXISTS `INF1343_A2`.`CUSTOMER` (  
  
`Customer_ID` CHAR(12) NOT NULL,  
  
`Preferred_Title` VARCHAR(45) NOT NULL,  
  
`Fname` VARCHAR(45) NOT NULL,  
  
`Minit` VARCHAR(45) NULL DEFAULT NULL,  
  
`Lname` VARCHAR(45) NOT NULL,  
  
`Address_Line` VARCHAR(100) NOT NULL,  
  
`Postal_Code` VARCHAR(6) NOT NULL,  
  
`City` VARCHAR(100) NOT NULL,  
  
`State` VARCHAR(50) NOT NULL,
```

```

`Country` VARCHAR(50) NOT NULL,
`EMPLOYEE_ID` CHAR(7) NOT NULL,
PRIMARY KEY (`Customer_ID`, `EMPLOYEE_ID`),
UNIQUE INDEX `ID_UNIQUE` (`Customer_ID` ASC) VISIBLE,
INDEX `fk_CUSTOMER_EMPLOYEE1_idx` (`EMPLOYEE_ID` ASC) VISIBLE,
CONSTRAINT `fk_CUSTOMER_EMPLOYEE1`
    FOREIGN KEY (`EMPLOYEE_ID`)
    REFERENCES `INF1343_A2`.`EMPLOYEE` (`Employee_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `INF1343_A2`.`PAYMENT`
-----

```

```

CREATE TABLE IF NOT EXISTS `INF1343_A2`.`PAYMENT` (
    `Order_ID` INT(4) NOT NULL,
    `Payment_Date` DATETIME NOT NULL,
    `Amount` FLOAT NOT NULL,
    `CUSTOMER_ID` CHAR(12) NOT NULL,
    `Method` VARCHAR(45) NOT NULL,
    `Payment_Status` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`Order_ID`, `CUSTOMER_ID`),
    UNIQUE INDEX `ID_UNIQUE` (`Order_ID` ASC) VISIBLE,
    INDEX `fk_PAYMENT_CUSTOMER1_idx` (`CUSTOMER_ID` ASC) VISIBLE,

```

```

CONSTRAINT `fk_PAYMENT_CUSTOMER1`

FOREIGN KEY (`CUSTOMER_ID`)

REFERENCES `INF1343_A2`.`CUSTOMER` (`Customer_ID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

-----

-- Table `INF1343_A2`.`CUSTOMER_ORDER`

-----

CREATE TABLE IF NOT EXISTS `INF1343_A2`.`CUSTOMER_ORDER` (

  `Order_ID` INT NOT NULL,

  `Customer_ID` CHAR(12) NOT NULL,

  `Order_Date` DATETIME NOT NULL,

  `Shipped_Date` DATETIME NOT NULL,

  `Comment` VARCHAR(800) NULL DEFAULT NULL,

  `Status` VARCHAR(45) NOT NULL,

  `Total_Price` FLOAT NOT NULL,

  PRIMARY KEY (`Order_ID`, `Customer_ID`),

  UNIQUE INDEX `ID_UNIQUE` (`Order_ID` ASC) VISIBLE,

  INDEX `Order_Payment_fk_idx` (`Order_ID` ASC, `Customer_ID` ASC) VISIBLE,

  CONSTRAINT `Customer ID`

    FOREIGN KEY (`Customer_ID`)

      REFERENCES `INF1343_A2`.`CUSTOMER` (`Customer_ID`)

    ON DELETE CASCADE

```



```
ON UPDATE CASCADE,
CONSTRAINT `Order_Payment_fk`
FOREIGN KEY (`Order_ID` , `Customer_ID`)
REFERENCES `INF1343_A2`.`PAYMENT` (`Order_ID` , `CUSTOMER_ID`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----
```

```
-- Table `INF1343_A2`.`PRODUCT`
```

```
-- -----
```

```
CREATE TABLE IF NOT EXISTS `INF1343_A2`.`PRODUCT` (
  `Product_Code` CHAR(9) NOT NULL,
  `Price` FLOAT NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Color` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`Product_Code`),
  UNIQUE INDEX `Code_UNIQUE` (`Product_Code` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----
```

```
-- Table `INF1343_A2`.`ORDER_ITEM`
```

```

-----

CREATE TABLE IF NOT EXISTS `INF1343_A2`.`ORDER_ITEM` (
  `Price_Each` FLOAT NOT NULL,
  `Quantity` INT NOT NULL,
  `Product_Code` CHAR(9) NOT NULL,
  `ORDER_ID` INT NOT NULL,
  INDEX `fk_ORDER_ITEM_ORDER1_idx` (`ORDER_ID` ASC) VISIBLE,
  PRIMARY KEY (`Product_Code`, `ORDER_ID`),
  CONSTRAINT `fk_ORDER_ITEM_ORDER1`
    FOREIGN KEY (`ORDER_ID`)
      REFERENCES `INF1343_A2`.`CUSTOMER_ORDER` (`Order_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Product Code`
    FOREIGN KEY (`Product_Code`)
      REFERENCES `INF1343_A2`.`PRODUCT` (`Product_Code`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----

-- Table `INF1343_A2`.`SHOPPING_CART`

```

```

-----

CREATE TABLE IF NOT EXISTS `INF1343_A2`.`SHOPPING_CART` (
  `Product_Code` CHAR(9) NOT NULL,

```

```

`Date_Added` DATETIME NOT NULL,

`Quantity` INT NOT NULL,

`CUSTOMER_ID` CHAR(12) NOT NULL,

PRIMARY KEY (`CUSTOMER_ID`),

UNIQUE INDEX `Product Code_UNIQUE` (`Product_Code` ASC) VISIBLE,

INDEX `fk_SHOPPING CART_CUSTOMER1_idx` (`CUSTOMER_ID` ASC) VISIBLE,

CONSTRAINT `fk_SHOPPING CART_CUSTOMER1`

    FOREIGN KEY (`CUSTOMER_ID`)

    REFERENCES `INF1343_A2`.`CUSTOMER` (`Customer_ID`)

    ON DELETE CASCADE

    ON UPDATE CASCADE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

```

```

-----

-- Table `INF1343_A2`.`Stock`

-----

CREATE TABLE IF NOT EXISTS `INF1343_A2`.`Stock` (

    `Quantity` INT NOT NULL,

    `PRODUCT_Code` CHAR(9) NOT NULL,

    INDEX `fk_Stock_PRODUCT1_idx` (`PRODUCT_Code` ASC) VISIBLE,

    UNIQUE INDEX `PRODUCT_Code_UNIQUE` (`PRODUCT_Code` ASC) VISIBLE,

    PRIMARY KEY (`PRODUCT_Code`),

    CONSTRAINT `fk_Stock_PRODUCT1`

        FOREIGN KEY (`PRODUCT_Code`)

        REFERENCES `INF1343_A2`.`PRODUCT` (`Product_Code`)

```

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

### **Insert data into table STOCK**

INSERT INTO Stock(Quantity, PRODUCT\_Code)

VALUES (67, 'MK1000001'),(38, 'CC1000001'), (63, 'SR1000001'),(50,  
'CS1000001'),(105,'MK1000002'),(50, 'CS1000002'),(20, 'LP1000001'),(18, 'LP1000002'),(19,  
'BP1000001');

Select \* from Stock;

### **Insert data into table PRODUCT**

INSERT INTO PRODUCT (Product\_Code,Price,Name,Color)

VALUES ('MK1000001', '2.99', 'Marker','Green'),('CC1000001', '26.99', 'Cloud Coffee Cup','Cream'),  
( 'SR1000001', '3.99', 'Sanrio Ruler','Light Blue'), ('CS1000001', '8.99', 'China  
Spoon','White'),('MK1000002', '2.99', 'Marker','Blue'),('CS1000002', '8.99', 'China  
Spoon','Silver'),('LP1000001', '39.99', 'Lamp','White'),('LP1000002', '39.99',  
'Lamp','Black'),('BP1000001', '46.99', 'Backpack','Yellow ');

Select \* from PRODUCT;

### **Insert data into table ORDER\_ITEM**

INSERT INTO ORDER\_ITEM (ORDER\_ID,Product\_Code,Price\_Each,Quantity)

VALUES ('1478','CC1000001', '26.99','2'),('1479','MK1000001', '2.99', '2'), ('1479','MK1000002',  
'2.99', '2'), ('1479','CS1000001', '8.99', '1'), ('1480','LP1000001', '39.99', '1'),('1480','MK1000002',  
'2.99', '3'),('1480','SR1000001', '3.99', '1'),('1480','BP1000001', '46.99', '1'),('1481','BP1000001',

```
'46.99', '1'),('1481','CS1000001', '8.99', '2'),('1481','CC1000001', '26.99', '1'),('1482','SR1000001', '3.99', '1'),('1482','LP1000002', '39.99', '2'),('1482','MK1000002', '2.99', '2');
```

```
Select * from ORDER_ITEM;
```

### **Insert data into table CUSTOMER\_ORDER**

Insert INTO

```
CUSTOMER_ORDER(Order_ID, Customer_ID, Order_Date, Shipped_Date, Comment, Status, Total_Price)
```

```
VALUES ('1478','SC2023000001','2023-01-30','2023-02-02','Make it a gift','Delivered','61.00'),('1479','CK2023000002','2023-02-02','2023-02-05','','Delivered','23.67'),('1480','JS2023000003','2023-02-15','2023-02-16','','Delivered','112.93'),('1481','XA2023000004','2023-03-01','2023-03-04','Need Packaging','Delivered','103.91'),('1482','AM2023000005','2023-03-12','2023-03-14','','Delivered','56.45');
```

```
Select * from CUSTOMER_ORDER;
```

### **Insert data into table PAYMENT**

Insert INTO

```
PAYMENT(Order_ID, Payment_Date, Amount, CUSTOMER_ID, Method, Payment_Status)
```

```
VALUES ('1478','2023-01-30','61.00','SC2023000001','Debit Card','Recieved'),('1479','2023-02-02','23.67','CK2023000002','Debit Card','Recieved'),('1480','2023-02-15','112.93','JS2023000003','Debit Card','Recieved'),('1481','2023-03-01','103.91','XA2023000004','Credit Card','Recieved'),('1482','2023-03-12','56.45','AM2023000005','Gift Card','Recieved');
```

```
Select * from PAYMENT;
```

### **Insert data into table Customer**

Insert INTO

```
CUSTOMER(Customer_ID, Preferred_Title, Fname, Minit, Lname, Address_Line, Postal_Code, City, State, Country, EMPLOYEE_ID)
```

```
VALUES ('SC2023000001','Mrs','Samantha','YD','Chen','4700 Keele Street','M4L0U8','North York','Ontario','Canada','2000001'),('CK2023000002','Miss','Chloe','JS','Kim','100 John West Way','L4E4E3','Aurora','Ontario','Canada','2000001'),('JS2023000003','Mr','Jackson','','Singh','1 Wellington Street','K1A0A3','Ottawa','Ontario','Canada','2000002'),('XA2023000004','Mr','Xavier','M','Adams','20
```

0 University Avenue Wes','H1K5E8','Waterloo','Ontario','Canada','2000003'),  
('AM2023000005','Miss','Ava','','Martinez','130 St. George  
Street','M4Y0E2','Toronto','Ontario','Canada','2000004');

Select \* from CUSTOMER;

### **Insert data into table SHIPPING\_CART**

INSERT INTO SHIPPING\_CART (Product\_Code,Date\_Added,Quantity,CUSTOMER\_ID)

VALUES ('CC1000001', '2023-03-07', '1','SC2023000001'),('SR1000001', '2023-03-10',  
'2','CK2023000002'),('BP1000001', '2023-03-14', '1','JS2023000003'),('CS1000001', '2023-03-20',  
'3','XA2023000004'),('MK1000002', '2023-03-26', '5','AM2023000005');

Select \* from SHIPPING\_CART;

### **Insert data into table BOSS**

Insert INTO BOSS(Boss\_ID,Fname,Minit,Lname,Phone\_Number,Email)

VALUES  
('1000001','Anthony','KY','Chou','4589137201','[Anthony.chou@outlook.com](mailto:Anthony.chou@outlook.com)'),('1000002','Summer','R  
L','Zhang','7126548109','Summer.zhang@outlook.com'),('1000003','Kiran','YR','Zheng','8315902483','  
yr9345.zheng@outlook.com'),('1000004','Samantha','','Nguyen','6041269378','Samantha.nguyen@outl  
ook.com'),('1000005','Gabrielle','','Rodriguez','2268896054','Gabri.rodriguez@outlook.com');

Select \* from BOSS;

### **Insert data into table EMPLOYEE**

Insert INTO

EMPLOYEE(Employee\_ID,SIN,Fname,Minit,Lname,Gender,Phone\_Number,Home\_Address,Email,  
BOSS\_ID)

VALUES ('2000001','425871639','Olivia ','W','Bennett','Female','6843907152','14 York Street,  
Toronto, ON ','[Olivia.W.Bennett@outlook.com](mailto:Olivia.W.Bennett@outlook.com)','1000001'),  
('2000002','639857124','Ethan','C','Nguyen','Male','2571469328','10 York Street, Toronto, ON  
,','[Ethan.C.Nguyen@outlook.com](mailto:Ethan.C.Nguyen@outlook.com)','1000004'),  
('2000003','210495837','Maya','','Patel','Female','9328407616','33 Charles Street, Toronto, ON  
,','[Maya.Patel@outlook.com](mailto:Maya.Patel@outlook.com)','1000002'),  
('2000004','978345216','Alex','B','Rodriguez','Male','7194638251','1080 Bay Street, Toronto, ON  
,','[Alex.B.Rodriguez@outlook.com](mailto:Alex.B.Rodriguez@outlook.com)','1000004'),  
('2000005','503286914','Lily','','Johnson','Female','5028315746','365 Dundas Street, Toronto, ON  
,','Lily.Johnson@outlook.com','1000005'), ('2000006','721694538','Sebastian

```
','Lee','Male','8539712860','1002 Yonge Street, Toronto, ON  
','Sebastian.Lee@outlook.com','1000003');
```

```
Select * from EMPLOYEE;
```

## View

**All products with their code, name, and price:**

```
CREATE VIEW Products_info AS
```

```
SELECT Product_Code, Name, Price FROM PRODUCT;
```

**All payments with their order ID, payment date, amount, and payment method:**

```
CREATE VIEW Payments_info AS
```

```
SELECT Order_ID, Payment_date, Amount, Method FROM PAYMENT;
```

**All employees with their name, email, and boss ID:**

```
CREATE VIEW Employees_boss AS
```

```
SELECT Fname,Minit,Lname, Email, Boss_ID FROM EMPLOYEE;
```

**All orders with their customer ID, order date, shipped date, and status:**

```
CREATE VIEW Orders_status AS
```

```
SELECT Customer_ID, Order_Date, Shipped_Date, status FROM CUSTOMER_ORDER;
```

**All shopping carts with their product code, quantity, and customer ID:**

```
CREATE VIEW Shopping_Carts_info AS
```

```
SELECT Product_Code, Quantity, Customer_ID FROM SHOPPING_CART;
```

**All orders with their customer ID, order date, and total price:**

```
CREATE VIEW Orders_total AS
```

```
SELECT Customer_ID, Order_Date, Total_Price FROM CUSTOMER_ORDER;
```

**All products with their code, name, color, and stock quantity:**

```
CREATE VIEW Products_stock AS
```

```
SELECT PRODUCT.Product_Code, PRODUCT.Name, PRODUCT.Color, STOCK.Quantity
```

```
FROM PRODUCT
```

```
INNER JOIN STOCK ON PRODUCT.Product_Code = STOCK.Product_Code;
```

**All customers with their name, address, city, state, and country:**

```
CREATE VIEW Customers_location AS
```

```
SELECT Customer_ID, Fname, Minit, Lname, Address_line, City, State, Postal_Code, Country  
FROM CUSTOMER;
```

**All employees with their SIN, name, and phone number:**

```
CREATE VIEW Employees_contact AS
```

```
SELECT SIN, Fname, Minit, Lname, Phone_Number FROM EMPLOYEE;
```

**All order items with their price, quantity, and product code:**

```
CREATE VIEW Order_items_info AS
```

```
SELECT Price_Each, Quantity, Product_Code FROM ORDER_ITEM;
```

## **Trigger:**

**A trigger that automatically updates the stock table whenever a new order is placed. This trigger will decrease the stock quantity of the products ordered by the quantity ordered.**

```
DELIMITER $$
```



```

CREATE TRIGGER update_stock_on_order

AFTER INSERT ON ORDER_ITEM

FOR EACH ROW

BEGIN

    UPDATE STOCK SET quantity = quantity - ORDER_ITEM.quantity WHERE Product_code =
ORDER_ITEM.Product_code;

END$$

```

DELIMITER ;

**Here's an example trigger to update the total price of a shopping cart whenever a new item is added or an existing item is updated:**

```

CREATE TRIGGER update_shopping_cart_total

AFTER INSERT OR UPDATE ON SHOPPING_CART

FOR EACH ROW

BEGIN

    UPDATE SHOPPING_CART

    SET total_price = (

        SELECT SUM(quantity * price)

        FROM PRODUCT

        WHERE SHOPPING_CART.product_code = PRODUCT.product_code

    )

    WHERE customer_id = NEW.customer_id;

END;

```

This trigger will calculate the total price for all items in the shopping cart belonging to the customer whose cart was updated (as specified by NEW.customer\_id). The SUM(quantity \* price) calculation multiplies the quantity of each item by its price, and then sums the total for all items in the shopping cart. The updated total is then set for the total\_price column in the SHOPPING\_CART table.

**Trigger to enforce a maximum quantity limit in the shopping cart:**

```

CREATE TRIGGER max_quantity BEFORE INSERT ON SHOPPING_CART
FOR EACH ROW
BEGIN
    DECLARE cart_quantity INT;

    SELECT IFNULL(SUM(quantity), 0) INTO cart_quantity
    FROM SHOPPING_CART
    WHERE customer_ID = NEW.customer_ID;

    IF (cart_quantity + NEW.quantity) > 10 THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Maximum quantity limit in shopping
cart exceeded';

    END IF;
END;

```

This trigger will prevent a customer from adding more than 10 items to their shopping cart by raising an error if the maximum quantity limit is exceeded.

**Trigger to automatically set the status of an order to "Shipped" when the shipped date is updated:**

```

CREATE TRIGGER update_order_status_to_shipped
AFTER UPDATE ON CUSTOMER_ORDER
FOR EACH ROW
BEGIN
    IF NEW.shipped_date IS NOT NULL AND NEW.status <> 'Shipped' THEN

        UPDATE CUSTOMER_ORDER

        SET status = 'Shipped'

        WHERE ID = NEW.ID;

    END IF;
END;

```

