MSIS2503 Database Management System – SQL

SQL Application: Business Scenario Analysis for H-Mart

Bingbing Pan

Table of Contents

Executive Summary

Project Summary

Firstly, I picked up a store to observe, after that, I wrote down the process of business operation and drew a swim lane diagram to present the business process. I then collected several receipts, obtained useful information from the receipt and recorded them in the excel spreadsheet in order to establish the database. After created the highly detailed table (denormalized table), I then created several normalized table in SQL so the business information can be stored appropriately. Then I used the dataset to analyze the business operation, such as top sales by customers and employee performance. Lastly, I used Tableau to create an interactive dashboard to visualize the answer of some business questions.

Business Scenario

A girl picks up a shopping cart and walks into H Mart located in Oakland Road, San Jose at 1:00 PM. She walks to the fruit section and picks up a box of cherries, two dragon fruits and a pearl. She then walks to the vegetable sections right next to the fruit section and picks up a box of mushrooms, a bag of bok choy, some green onions, and a bag of sweet potatoes. She later walks to the diary sections, but she cannot find the yogurt she wants, therefore, she goes to the service desk to ask whether the yogurt is in-stock. After a while, the store manager first checks his computer for inventory status, then brings the yogurt she wants, and put more in the fridge. She then walks to the meat section, and picks up some hotpot beefs and hotpot lambs, it seems like she is going to have hotpot later. She then walks to aisle 3 for the hotpot base, however, she cannot find what she wants. She goes to the service desk again, and ask the store manager about whether the hotpot base is in-stock. After the store manager checks the store inventory, he tells the girl that the hotpot base is out of stock. So the girl returns to aisle 3 and pick up the other brand's hotpot base. She then walks to the beverage section, which is aisle 4 and picks up half dozen of Ceylon Lemon Tea Drink, and a big bottle of barley tea. After that, she walks to the checkout station 5 to check out. The store employee asks her whether she has the membership card, and the girl said no, and the store employee asks if she want to issue a membership card so she can accumulate credits for future discounts, the girl still says no. The store employee then asks her if she needs bags, she said yes, she needs two bags, then the store employee scans the barcode for the bags for twice. After that, the store employee starts scanning all the things in her shopping cart, then mentions to her that it is a total of $126.54 including tax. The girl uses Apple Pay to make the payment. The girl takes receipt from the store employee and walks to her car in parking lot at 1:37 PM.

Attached is the swim lane diagram (Chart 1) to summarize the business process:

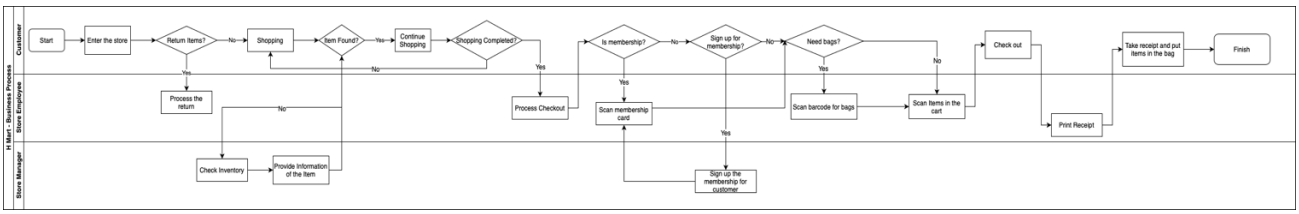

Chart 1

Dataset Creation

Denormalized Table Creation

I obtained useful information from store receipts, which includes customer ID, membership information, product ID, product name, product location, product price, product type, product quantity, product unit, order ID, order timestamp, order balance, employee ID. I also gather other information from the store manager. Then I created the sample dataset by putting all the information in the table. Attached is the sample dataset (Table 1).

CustomerID	First_Name	Last_Name	Gender	Location_City	Age	Memebership	ProductID	Product_Name	Product_Type	Product_Price	Product_Quantity	Product_Unit	Product_Location	Order_ID	Order_Timestamp	Order_Balance	EmployeeID	Employee_FirstName	Employee_LastName	Employee_Position	Shopping_Experience
1020	Spencer	Little	F	San Jose	18	Yes	VG1020	Spinach	Vegetable	2.34	1.5	LB	1	113	2/13/20 11:01:03	33.01	2251	Emillie	Dunlap	Senior Check-out	4.8
1020	Spencer	Little	F	San Jose	18	Yes	FR2209	Dragon Fruit	Fruit	4.19	0.8	LB	3	113	2/13/20 11:01:03	33.01	2251	Emillie	Dunlap	Senior Check-out	4.8
1020	Spencer	Little	F	San Jose	18	Yes	GE3879	Pot	General	23.99	1.0	Piece	12	113	2/13/20 11:01:03	33.01	2251	Emillie	Dunlap	Senior Check-out	4.8
1021	Aadam	Croft	M	San Jose	20	No	VG1029	Potato	Vegetable	1.36	2	LB	2	114	2/13/20 11:03:05	25.04	2256	Brittney	Barnes	Check-out	3.2
1021	Aadam	Croft	M	San Jose	20	No	VG1556	Tomato	Vegetable	1.45	1.3	LB	2	114	2/13/20 11:03:05	25.04	2256	Brittney	Barnes	Check-out	3.2
1021	Aadam	Croft	M	San Jose	20	No	FR2209	Dragon Fruit	Fruit	4.19	1.8	LB	3	114	2/13/20 11:03:05	25.04	2256	Brittney	Barnes	Check-out	3.2
1021	Aadam	Croft	M	San Jose	20	No	SN7765	Dove Chocolate	Snack	3.99	1.0	Piece	6	114	2/13/20 11:03:05	25.04	2256	Brittney	Barnes	Check-out	3.2
1021	Aadam	Croft	M	San Jose	20	No	DR9876	Coke	Drink	8.00	1.0	Dozen	7	114	2/13/20 11:03:05	25.04	2256	Brittney	Barnes	Check-out	3.2
1022	Noah	William	M	Mountain View	36	No	VG1029	Potato	Vegetable	1.36	1.8	LB	2	115	2/13/20 11:05:14	20.48	2254	Dawud	Mahoney	Check-out	4.2
1022	Noah	William	M	Mountain View	36	No	VG1020	Spinach	Vegetable	2.34	1.2	LB	1	115	2/13/20 11:05:14	20.48	2254	Dawud	Mahoney	Check-out	4.2
1022	Noah	William	M	Mountain View	36	No	FR2278	Lemon	Fruit	2.50	2.3	LB	3	115	2/13/20 11:05:14	20.48	2254	Dawud	Mahoney	Check-out	4.2
1022	Noah	William	M	Mountain View	36	No	DR9876	Coke	Drink	8.00	1.0	Dozen	7	115	2/13/20 11:05:14	20.48	2254	Dawud	Mahoney	Check-out	4.2
1022	Noah	William	M	Mountain View	36	No	VG8798	Garlic	Vegetable	1.25	0.6	LB	1	115	2/13/20 11:05:14	20.48	2254	Dawud	Mahoney	Check-out	4.2
1023	Buster	Melendez	M	Sunnyvale	32	Yes	VG8798	Garlic	Vegetable	1.25	0.7	LB	1	116	2/13/20 11:09:23	7.42	2251	Emillie	Dunlap	Senior Check-out	5
1023	Buster	Melendez	M	Sunnyvale	32	Yes	SN8756	Ice Cream	Snack	6.00	1.0	Piece	15	116	2/13/20 11:09:23	7.42	2251	Emillie	Dunlap	Senior Check-out	5
1024	Raheem	Coleman	M	Los Gatos	30	Yes	VG8798	Garlic	Vegetable	1.25	0.8	LB	1	117	2/13/20 11:10:56	12.09	2256	Brittney	Barnes	Check-out	4.6
1024	Raheem	Coleman	M	Los Gatos	30	Yes	FD9089	Rice Cake	Staple	7.89	1.0	Bag	14	117	2/13/20 11:10:56	12.09	2256	Brittney	Barnes	Check-out	4.6
1024	Raheem	Coleman	M	Los Gatos	30	Yes	VG3366	Cucumber	Vegetable	2.49	1.0	LB	2	117	2/13/20 11:10:56	12.09	2256	Brittney	Barnes	Check-out	4.6
1025	Honor	Levy	F	Santa Clara	57	Yes	VG8799	Ginger	Vegetable	1.25	0.6	LB	1	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1025	Honor	Levy	F	Santa Clara	57	Yes	VG3366	Cucumber	Vegetable	2.49	1.3	LB	1	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1025	Honor	Levy	F	Santa Clara	57	Yes	FR2278	Lemon	Fruit	2.50	2.0	LB	3	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1025	Honor	Levy	F	Santa Clara	57	Yes	VG7654	Bok Choy	Vegetable	2.44	1.6	LB	2	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1025	Honor	Levy	F	Santa Clara	57	Yes	VG8789	Red Pepper	Vegetable	2.80	0.8	LB	2	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1025	Honor	Levy	F	Santa Clara	57	Yes	GE3431	Toothpaste	General	3.99	1.0	Piece	13	118	2/13/20 11:13:34	19.48	2254	Dawud	Mahoney	Check-out	4.5
1026	Evelyn	Bellamy	F	Sunnyvale	55	No	VG1556	Tomato	Vegetable	1.45	1.5	LB	2	119	2/13/20 11:16:21	6.45	2251	Emillie	Dunlap	Senior Check-out	4.5
1026	Evelyn	Bellamy	F	Sunnyvale	55	No	VG1020	Spinach	Vegetable	2.34	1.4	LB	1	119	2/13/20 11:16:21	6.45	2251	Emillie	Dunlap	Senior Check-out	4.5
1026	Evelyn	Bellamy	F	Sunnyvale	55	No	CG8798	Garlic	Vegetable	1.25	0.8	LB	1	119	2/13/20 11:16:21	6.45	2251	Emillie	Dunlap	Senior Check-out	4.5
1027	Krish	Person	M	Mountain View	43	Yes	VG3366	Cucumber	Vegetable	2.49	2.3	LB	1	120	2/13/20 11:17:45	20.52	2256	Brittney	Barnes	Check-out	5
1027	Krish	Person	M	Mountain View	43	Yes	FR2278	Lemon	Fruit	2.50	3.3	LB	3	120	2/13/20 11:17:45	20.52	2256	Brittney	Barnes	Check-out	5
1027	Krish	Person	M	Mountain View	43	Yes	SN8756	Ice Cream	Snack	6.00	1.0	Piece	15	120	2/13/20 11:17:45	20.52	2256	Brittney	Barnes	Check-out	5
1028	Gabija	Vang	M	Santa Clara	28	No	FD9089	Rice Cake	Staple	7.89	1	Bag	14	121	2/13/20 11:19:47	8.6	2254	Dawud	Mahoney	Check-out	5
1029	Haider	Muir	F	San Jose	19	No	VG1556	Tomato	Vegetable	1.45	2.5	LB	2	122	2/13/20 11:20:53	12.35	2251	Emillie	Dunlap	Senior Check-out	5
1029	Haider	Muir	F	San Jose	19	No	DR9876	Coke	Drink	8.00	1	Dozen	7	122	2/13/20 11:20:53	12.35	2251	Emillie	Dunlap	Senior Check-out	5

Table 1

Normalized Table Creation

After putting all the information into the spreadsheet, I then created 7 normalized tables (Exhibit 1-1 to Exhibit 1-7) in MySQL workbench. The purpose of creating normalized table is to validate and improve a logical design of the database so that it satisfies certain constraints that avoid unnecessary duplication of data. Moreover, normalized tables contain minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies. Later, I will use the normalized tables to answer some business questions.

```
CREATE TABLE IF NOT EXISTS Customer (
    CustomerID INT NOT NULL,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Gender VARCHAR(1) NOT NULL,
    Location_City VARCHAR(100) NOT NULL,
    Age INT NOT NULL,
    CONSTRAINT PKCustomerID PRIMARY KEY(CustomerID)
);
```

Exhibit 1-1

```
CREATE TABLE IF NOT EXISTS Employee (
    EmployeeID INT NOT NULL,
    Employee_FirstName VARCHAR(50) NOT NULL,
    Employee_LastName VARCHAR(50) NOT NULL,
    Employee_Position VARCHAR(50) NOT NULL,
    CONSTRAINT PKEmployeeID PRIMARY KEY(EmployeeID)
);
```

Exhibit 1-2

```
CREATE TABLE IF NOT EXISTS Product (
    ProductID VARCHAR(20) NOT NULL,
    Product_Name VARCHAR(50) NOT NULL,
    Product_Type VARCHAR(50) NOT NULL,
    Product_Price NUMERIC(5, 3) NOT NULL,
    Product_Unit VARCHAR(20) NOT NULL,
    Product_Location INT,
    CONSTRAINT PKProductID PRIMARY KEY(ProductID)
);
```

Exhibit 1-3

```
CREATE TABLE IF NOT EXISTS Membership (
    CustomerID INT NOT NULL,
    Memebership VARCHAR(10) NOT NULL,
    CONSTRAINT FK1 FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);
```

Exhibit 1-4

```
CREATE TABLE IF NOT EXISTS Order_info (
    Order_ID INT NOT NULL,
    Order_Timestamp TIMESTAMP,
    Order_Balance NUMERIC(5, 3) NOT NULL,
    CONSTRAINT PKOrder_ID PRIMARY KEY(Order_ID)
);
```

Exhibit 1-5

```
CREATE TABLE IF NOT EXISTS Shopping_Experience (
    CustomerID INT,
    Shopping_Experience NUMERIC(2, 1),
    CONSTRAINT FK2 FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);
```

Exhibit 1-6

```
CREATE TABLE IF NOT EXISTS Customer_Shopping (
    CustomerID INT NOT NULL,
    ProductID VARCHAR(20) NOT NULL,
    Product_Quantity NUMERIC(10, 2) NOT NULL,
    Order_ID INT NOT NULL,
    EmployeeID INT NOT NULL,
    CONSTRAINT FK3 FOREIGN KEY (CustomerID) REFERENCES  Customer (CustomerID),
    CONSTRAINT FK4 FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
    CONSTRAINT FK5 FOREIGN KEY (Order_ID) REFERENCES Order_info (Order_ID),
    CONSTRAINT FK6 FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID)
);
```

Exhibit 1-7

Relationship Between Each Normalized Tables

The ER diagram (Exhibit 2) attached shows the contents of each table and relationship between then. Customer table is the parent table of membership, shopping_experience, and customer_shopping tables. The customer table has the primary key of CustomerID, which is the foreign keys for membership, shopping_experience, and customer_shopping tables, so these 4 tables can logically connected together. Meanwhile, customer_shopping is the parent table of employee, order_info, and product tables. They are logically connected through customerID, employeeID, order_id, and productID. I will use these foreign keys to join the tables together later to answer the business questions.

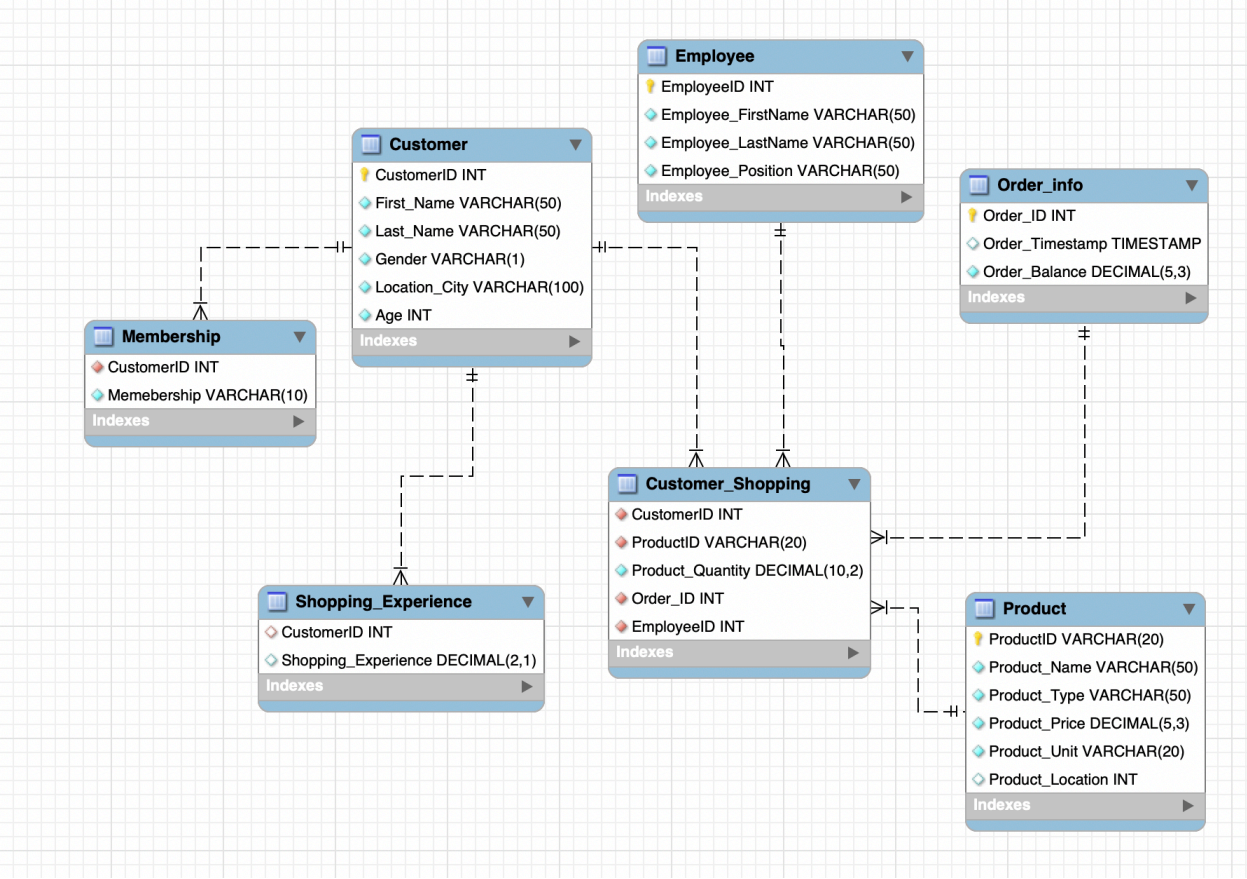

Exhibit 2

SQL Statement Creation

SQL statements are divided into five different categories: Data definition language (DDL), Data manipulation language (DML), Data Control Language (DCL), Transaction Control Statement (TCS), Session Control Statements (SCS). In the normalized table creation section, I use data definition language, such as CREATE syntax to create tables. In this section, I will use data manipulation language, such as SELECT, INSERT syntax to perform the business analysis.

Basic Case

Based on the table created above, SQL queries and perform analysis to answer some basic business question and create views for easier data extraction. In the following, I will show 5 basic SQL queries and 2 views using Oracle SQL.

SQL Queries Without Join

1. The business question here is finding out what's the average amount of each order. By answering the question, company can easily estimate the daily, monthly, even yearly sales. In attached query (Exhibit 3-1), I use the AVG syntax to get the average sales of each order.

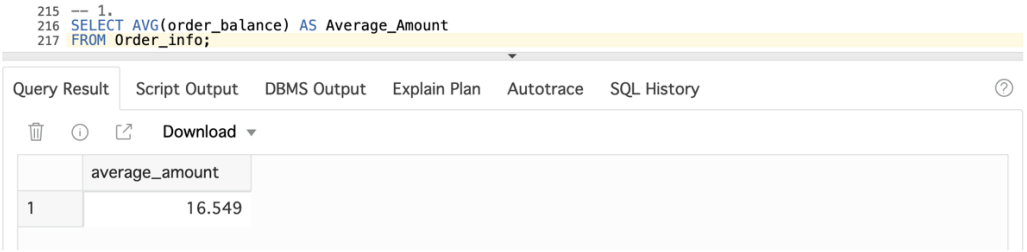

Exhibit 3-1

2. The business question here is finding out how many customers have membership. By answering this question, company can identify the number of royal customers. In attached query (Exhibit 3-2), I first use COUNT syntax to get the total number of customers, and then I use WHERE syntax to filter out customers who do not membership.

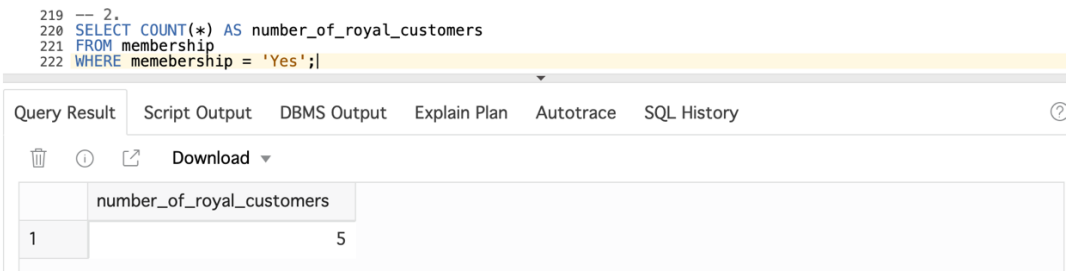

Exhibit 3-2

3. The business question here is finding out how many customers are females. By answering this question, company can identify the demographic information of its customers to better perform marketing decisions. In attached query (Exhibit 3-3), I first use COUNT syntax to get the total number of customers, and then I use WHERE syntax to extract customers who are females.

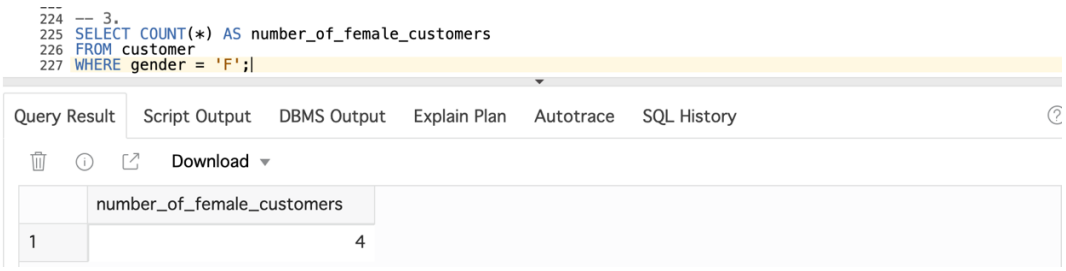

Exhibit 3-3

4. The business question here is finding out how many customers are in the age range of 20-40. By answering this questions, company can identify the demographic information of its customers, and perform its marketing campaign accordingly. In attached query (Exhibit 3-4), I first use COUNT syntax to get the total number of customers, and then I use WHERE syntax paired with AND syntax to give the condition, so the query will extract customers who are in the age range of 20-40.

```
229 -- 4.
230 SELECT COUNT(*) AS twenty_to_fourty
231 FROM customer
232 WHERE age >= 20 AND age <= 40;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	twenty_to_fourty
1	5

Exhibit 3-4

5. The business question here is finding out the number of customers from different cities. By doing so, company can identify customer's geographical information, and decide the heavy marketing region. In attached query (Exhibit 3-5), I first use SELECT syntax to extract the city names, and then I use COUNT syntax to get the total number of customers, I use GROUP BY syntax in the end to group the customers by cities, so the number of each city's customer will be showed accordingly.

```
234 -- 5.
235 SELECT location_city, COUNT(*) AS number_of_residents
236 FROM customer
237 GROUP BY location_city;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	location_city	number_of_residents
1	Los Gatos	1
2	Sunnyvale	2
3	Mountain View	2
4	Santa Clara	2
5	San Jose	3

Exhibit 3-5

Views Creation

1. Since normalized tables highly decentralize data, membership information and corresponding customer information are in two separated tables. If the manager barely knows SQL JOIN syntax but wants to check royal customer information, creating a view in SQL for the manager will be very convenient. The following SQL query (Exhibit 4-1) can create the view to store royal customer information.

```
124 CREATE VIEW RoyalCustomer AS
125 SELECT c.customerid, c.gender, c.age
126 FROM membership m
127 JOIN customer c
128 ON m.customerid = c.customerid
129 WHERE m.memebership = 'Yes';
130
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

```
View ROYALCUSTOMER created.
```

Exhibit 4-1

The manager can simply call the following syntax (Exhibit 4-2) to see the royal customer information.

```
131 SELECT *
132 FROM RoyalCustomer;
133
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	customerid	gender	age
1	1020	F	18
2	1023	M	32
3	1024	M	30
4	1025	F	57
5	1027	M	43

Exhibit 4-2

2. The inventory department usually needs to now popular products in order to better perform inventory management. By creating the view (Exhibit 4-3), inventory department can easily know how many product has been sold, when to restock the inventories, and what inventories level needs to be reach.

```
130 -- 2.
131 CREATE VIEW PopularProducts AS
132 SELECT p.product_name, p.product_type, p.product_price, p.product_unit, COUNT(*) AS number_of_purchase
133 FROM product p
134 JOIN customer_shopping cs
135 ON p.productid = cs.productid
136 GROUP BY p.product_name, p.product_type, p.product_price, p.product_unit
137 ORDER BY COUNT(*) DESC
138 FETCH NEXT 5 ROWS ONLY;
139
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

```
View POPULARPRODUCTS created.
```

Exhibit 4-3

The inventory can easily call the following syntax (Exhibit 4-4) to check the result.

```
140 SELECT *
141 FROM PopularProducts;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	product_name	product_type	product_price	product_unit	number_of_purchase
1	Garlic	Vegetable	1.25	LB	4
2	Cucumber	Vegetable	2.49	LB	3
3	Tomato	Vegetable	1.45	LB	3
4	Spinach	Vegetable	2.34	LB	3
5	Lemon	Fruit	2.5	LB	3

Exhibit 4-4

Advanced Case

Since the table normalization process high decentralized information into several different tables, a lot of business questions cannot be answer with the information in a single table. Therefore, in order to better serve business, syntax such as JOIN, TRIGGER, and PROCEDURE is required. In the following section, I will show 10 SQL queries with JOIN, 1 TRIGGER creation, and 1 PROCEDURE creation examples.

SQL Queries With Joins

1. The business question here is finding out top 5 customers who have the highest amount purchase. By answering the question, company can send promotions to those customers, and encourage them to buy more in the future. In the attached query (Exhibit 5-1), I join order_info table and customer table together in order to extract the needed information.

```
241 -- 1.
242 SELECT DISTINCT c.customerid, c.first_name, c.last_name, o.order_balance
243 FROM customer c
244 JOIN customer_shopping cs
245 ON c.customerid = cs.customerid
246 JOIN order_info o
247 ON cs.order_id = o.order_id
248 ORDER BY o.order_balance DESC
249 FETCH NEXT 5 ROWS ONLY;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	customerid	first_name	last_name	order_balance
1	1020	Spencer	Little	33.01
2	1021	Aadam	Croft	25.04
3	1027	Krish	Person	20.52
4	1022	Noah	William	20.48
5	1025	Honor	Levy	19.48

Exhibit 5-1

2. The business question here is finding out the average rating for each employee. By answering the question, the HR department can identify each employee's customer service performance. In the attached query (Exhibit 5-2), I join employee, customer_shopping, and shopping_experience tables together in order to extract the needed information, and sort the result by their average rating.

```
251 -- 2.
252 SELECT e.employeeid, e.employee_firstname, e.employee_lastname, AVG(s.shopping_experience) AS
    average_rating
253 FROM employee e
254 JOIN customer_shopping cs
255 ON e.employeeid = cs.employeeid
256 JOIN shopping_experience s
257 ON cs.customerid = s.customerid
258 GROUP BY e.employeeid, e.employee_firstname, e.employee_lastname
259 ORDER BY AVG(s.shopping_experience);
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	employeeid	employee_firstname	employee_lastname	average_rating
1	2256	Brittney	Barnes	4.07272727272727
2	2254	Dawud	Mahoney	4.41666666666666
3	2251	Emillie	Dunlap	4.79

Exhibit 5-2

3. The business question here is finding out what gender has greater shopping power. By answering the question, company can send out more promotion to the certain gender of people in order to achieve better marketing result. In the attached query (Exhibit 5-3), I join customer, customer_shopping, and order_info tables together in order to extract the needed information, and sort the result by the total balance in the end.

```
261 -- 3.
262 SELECT c.gender, SUM(o.order_balance) AS total_balance
263 FROM customer c
264 JOIN customer_shopping cs
265 ON c.customerid = cs.customerid
266 JOIN order_info o
267 ON cs.order_id = o.order_id
268 GROUP BY c.gender
269 ORDER BY SUM(o.order_balance);
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	gender	total_balance
1	F	519.92
2	M	697.84

Exhibit 5-3

4. The business question here is finding out whether customers with membership have more shopping power. By answering the question, company can whether membership program actually help with its sales. In the attached query (Exhibit 5-4), I join membership, customer_shopping,

and order_info tables together in order to extract the needed information, and sort the result by the total balance in the end.

```
-- 4.
SELECT m.memebership, SUM(o.order_balance) AS total_balance
FROM membership m
JOIN customer_shopping cs
ON m.customerid = cs.customerid
JOIN order_info o
ON cs.order_id = o.order_id
GROUP BY m.memebership;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	memebership	total_balance
1	No	1121.2
2	Yes	1314.32

Exhibit 5-4

5. The business question here is finding out which city has more shopping power. By answering the question, company can decide where to send out more promotion. In the attached query (Exhibit 5-5), I join customer, customer_shopping, and order_info tables together in order to extract the needed information.

```
-- 5.
SELECT c.location_city, SUM(order_balance) AS total_balance
FROM customer c
JOIN customer_shopping cs
ON c.customerid = cs.customerid
JOIN order_info o
ON cs.order_id = o.order_id
GROUP BY c.location_city;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download | Execution time: 0.073 seconds

	location_city	total_balance
1	Los Gatos	36.27
2	Sunnyvale	34.19
3	Mountain View	163.96
4	Santa Clara	125.53
5	San Jose	248.93

Exhibit 5-5

6. The question here is finding out the top 5 most popular product. By answering this question, company can do better on its inventory management by knowing the product sales level. In the attached query (Exhibit 5-6), I join product, and customer_shopping tables together in order to extract the needed information. Then I use GROUP BY syntax to group the result by product ID and product name, and sort the result by the total product quantity in the descending way. Lastly, I limit the showing result to 5, so only top 5 result will be showed.

```
289 -- 6.
290 SELECT p.productid, p.product_name, SUM(cs.product_quantity) AS total_quantity
291 FROM product p
292 JOIN customer_shopping cs
293 ON p.productid = cs.productid
294 GROUP BY p.productid, p.product_name
295 ORDER BY SUM(cs.product_quantity) DESC
296 FETCH NEXT 5 ROWS ONLY;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	productid	product_name	total_quantity
1	FR2278	Lemon	15.2
2	VG1556	Tomato	10.6
3	VG3366	Cucumber	9.2
4	VG1020	Spinach	8.2
5	VG1029	Potato	7.6

Exhibit 5-6

7. The business question here is finding out the top 5 popular product type. By answering this question, company can do better on its inventory management by knowing the product type sales level. In the attached query (Exhibit 5-7), I join product, and customer_shopping tables together in order to extract the needed information. I then use GROUP BY syntax to group the result by product type, and sort the result by the total product quantity in the descending way. Lastly, I limit the showing result to 5, so only top 5 result will be showed.

```
84 -- 7.
85 SELECT p.product_type, SUM(cs.product_quantity) AS total_quantity
86 FROM product p
87 JOIN customer_shopping cs
88 ON p.productid = cs.productid
89 GROUP BY p.product_type
90 ORDER BY SUM(cs.product_quantity) DESC
91 FETCH NEXT 5 ROWS ONLY;
92
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	product_type	total_quantity
1	Vegetable	47.4
2	Fruit	20.4
3	Drink	6
4	Snack	6
5	General	4

Exhibit 5-7

8. The question here is finding out the age range with the greater shopping power. By answering this question, company can send out promotions based on customers' age. In the attached query (Exhibit 5-8), I join customer, and customer_shopping tables together in order to extract the needed information. I then use GROUP BY syntax to group the result by age, and sort the result by the total sales in the descending way.

```
93  -- 8.
94  SELECT c.age, SUM(o.order_balance) AS total_balance
95  FROM customer c
96  JOIN customer_shopping cs
97  ON c.customerid = cs.customerid
98  JOIN order_info o
99  ON cs.order_id = o.order_id
100 GROUP BY c.age
101 ORDER BY SUM(o.order_balance) DESC;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	age	total_balance
1	20	250.4
2	57	233.76
3	36	204.8
4	18	198.06
5	43	123.12

Exhibit 5-8

9. The question here is finding out each employee's sales. By answering this question, company can identify which employee make the most sales, which is a key factor to evaluate employee performance. In the attached query (Exhibit 5-9), I join employee, customer_shopping, and order_info tables together in order to extract the needed information. I then use GROUP BY syntax to group the result by employee ID, employee's name, and sort the result by the total sales in the descending way.

```
103 -- 9.
104 SELECT e.employeeid, e.employee_firstname, e.employee_lastname, SUM(o.order_balance) AS total_balance
105 FROM employee e
106 JOIN customer_shopping cs
107 ON e.employeeid = cs.employeeid
108 JOIN order_info o
109 ON cs.order_id = o.order_id
110 GROUP BY e.employeeid, e.employee_firstname, e.employee_lastname
111 ORDER BY SUM(o.order_balance) DESC;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	employeeid	employee_firstname	employee_lastname	total_balance
1	2254	Dawud	Mahoney	455.86
2	2256	Brittney	Barnes	446.06
3	2251	Emillie	Dunlap	315.84

Exhibit 5-9

10. Finding out which aisle customer visit the most often. By doing so, company can find out which aisle's product should be refilled more often. In the attached query (Exhibit 5-10), I join product, and customer_shopping tables together in order to extract the needed information. I then use GROUP BY syntax to group the result by product location, and sort the result by the total visit times in the descending way.

```
113 -- 10.
114 SELECT p.product_location, COUNT(*) AS times_of_visit
115 FROM product p
116 JOIN customer_shopping cs
117 ON cs.productid = p.productid
118 GROUP BY p.product_location
119 ORDER BY COUNT(*) DESC;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	product_location	times_of_visit
1	2	20
2	1	16
3	3	10
4	7	6
5	14	4

Exhibit 5-10

Trigger and Procedure

When the company has new hires, it would be relatively complicated to perform ALTER TABLE syntax to add records into the dataset. Therefore, I create a trigger to update employee record if there's any new hire. In order to create the trigger, I first create a new table - EmployeeInsert (Exhibit 6-1). Secondly, I define an insert trigger (Exhibit 6-2). Thirdly, I try to insert a new record in the employee table (Exhibit 6-3). Fourthly, I check if the trigger works (Exhibit 6-4). Lastly, I check if the record got inserted into the table (Exhibit 6-5).

```
5  CREATE TABLE EmployeeInsert (
6    EmployeeID INT NOT NULL,
7    Employee_FirstName VARCHAR(50) NOT NULL,
8    Employee_LastName VARCHAR(50) NOT NULL,
9    Employee_Position VARCHAR(50) NOT NULL);
```

100% 41:9

Action Output

	Time	Action	Response	Duration / Fetch Time
1	17:41:06	CREATE TABLE Employe...	0 row(s) affected	0.029 sec

Exhibit 6-1

```
11   -- Define an insert trigger
12   Create Trigger EmployeeInsertTrigger AFTER INSERT ON Employee
13   FOR EACH ROW
14   INSERT INTO EmployeeInsert
15   SELECT EmployeeID, Employee_FirstName, Employee_LastName, Employee_Position
16   FROM Employee;
17
18
19
```

100% 15:16

Action Output

	Time	Action	Response	Duration / Fetch Time
1	17:44:07	Create Trigger Employe...	0 row(s) affected	0.012 sec

Exhibit 6-2

```
18   -- Try to insert a new record in the employee table
19   INSERT INTO Employee(EmployeeID, Employee_FirstName, Employee_LastName, Employee
20   VALUES(2257,'David','Richard', 'Manager');
```

100% 43:20

Action Output

	Time	Action	Response	Duration / Fetch Time
1	17:46:08	INSERT INTO Employee(...	1 row(s) affected	0.013 sec

Exhibit 6-3

```
22   -- Check if the trigger worked
23   SELECT * FROM EmployeeInsert;
```

100% 30:23

Result Grid | Filter Rows: Search | Export:

EmployeeID	Employee_FirstName	Employee_LastName	Employee_Position
2251	Emillie	Dunlap	Senior Check-out
2254	Dawud	Mahoney	Check-out
2256	Brittney	Barnes	Check-out
2257	David	Richai	

Result Grid | Form Editor

Exhibit 6-4

```
25   -- Check if the record got inserted into the table
26   SELECT * FROM Employee;
```

100% 24:26

Result Grid | Filter Rows: Search | Edit: | Export/Import: | Result Grid | Form Editor

EmployeeID	Employee_FirstName	Employee_LastName	Employee_Position
2251	Emillie	Dunlap	Senior Check-out
2254	Dawud	Mahoney	Check-out
2256	Brittney	Barnes	Check-out
2257	David	Richard	Manager
NULL	NULL	NULL	NULL

Exhibit 6-5

What if the company want to check the sales made by customers who have membership versus the sales made by customers who does not have membership? Instead of writing several sub queries to find out the answer, simply call out a procedure can help the company to find out the answer. In order to do so, I define the stored procedure (Exhibit 6-6). Then, I call each procedure to test if the procedure works (Exhibit 6-7, Exhibit 6-8).

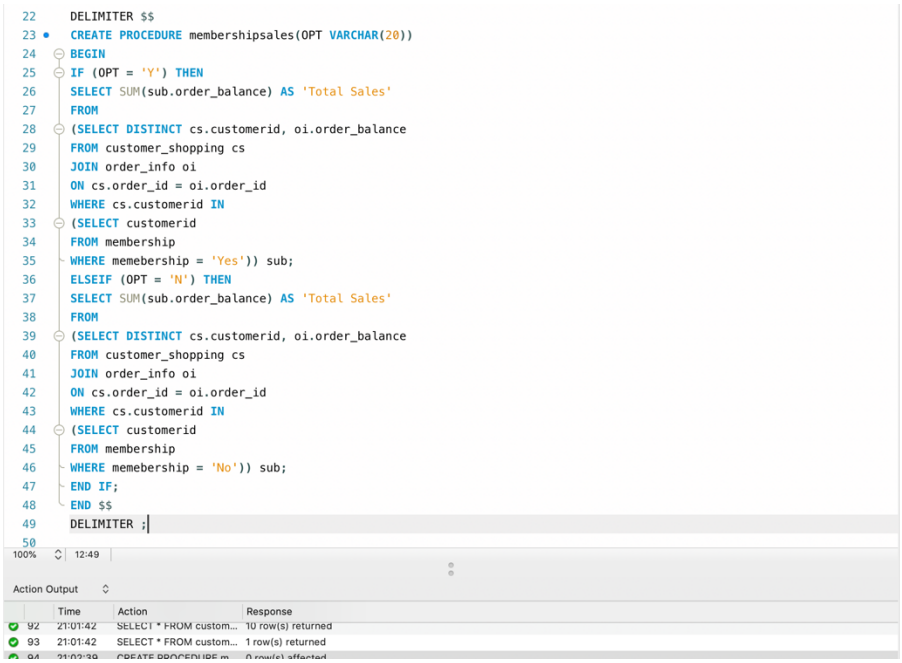

Exhibit 6-6

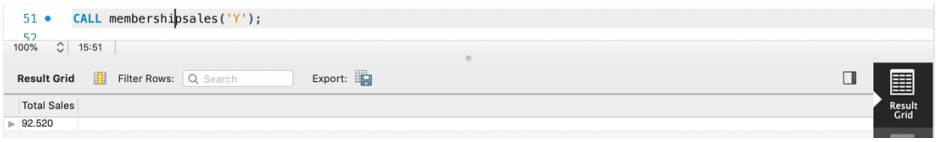

Exhibit 6-7

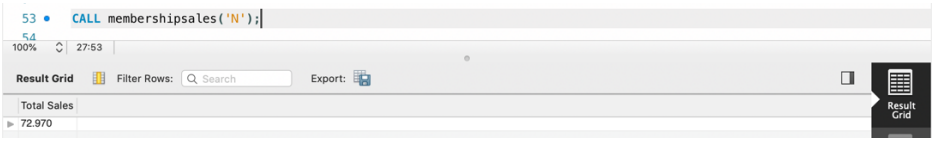

Exhibit 6-8

Analytical Result Visualization

Data visualization is one of the most effective way to present the data analytical result to people with limit data analytics knowledge. I use Tableau to visualize the dataset in different perspectives, such as to 10 customers with the largest purchase, sales ranking by product type, member's total sales vs non-member total sale, and average employee ratings. After creating different graphs, I put them all in a dashboard (Exhibit 7) with a filter on the right, by adjusting the filter, the graphs will have interactive function.

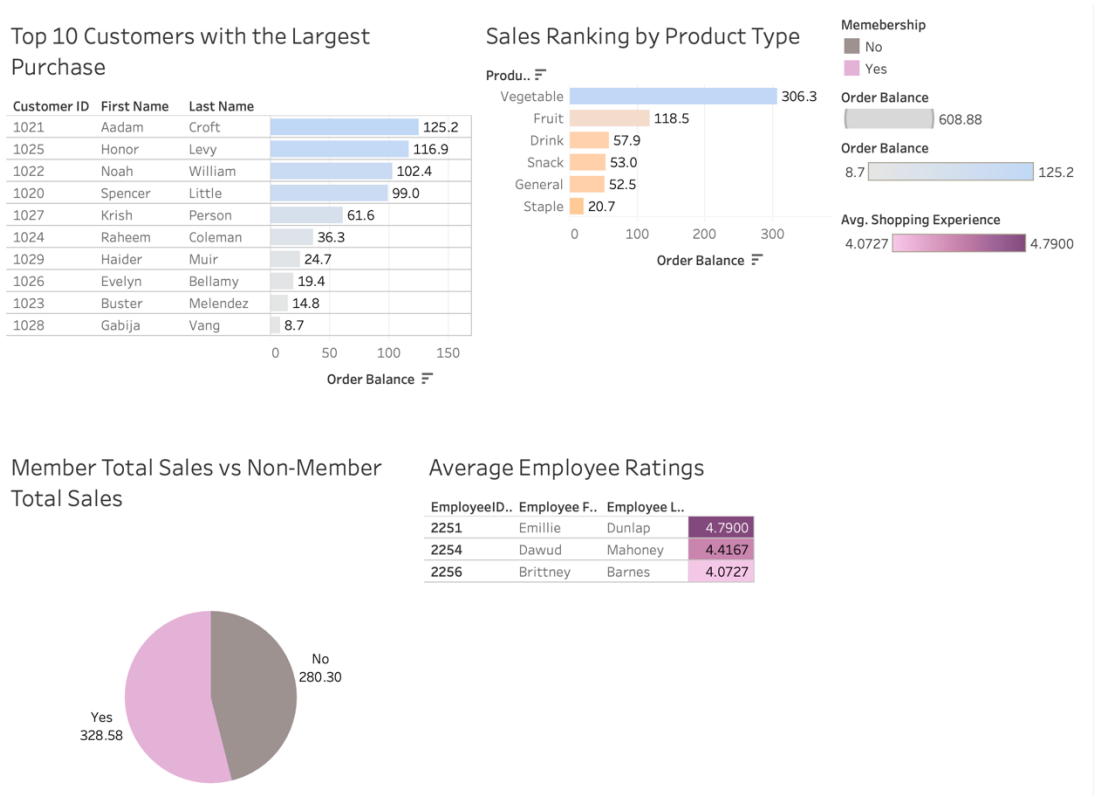

Exhibit 7

Limitation

With the limited information in the dataset, it's really difficult to perform an in-depth analysis, and some of the business questions might not be needed as often in the real work. However, the syntaxes performed in the project should be widely used in the real business problems. The queries should work perfectly in large dataset.

Conclusion

In this project, I use both MySQL and Oracle SQL to create tables, write SQL queries to solve different business questions, and use Tableau to visualize the analytical result. After this project, it's really important for me to learn that MySQL and Oracle SQL have really different syntax. In order to improve the business process, I think putting instruction board in front of each aisle to show what aisle has what type of product would make customers shopping experience more enjoyable.