Module Code: CT087-0-M

Module Name: Database for Data Science

Student's TP: TP075628

Student's Name: Abigail Wee Tynn

Lecturer's Name: Ms. Lai Chew Ping

Table of Contents

1.0 Introduction

This report aims to document the design and implementation of a database system for an electronic bookstore called E-Bookstore. The report consists of an introduction where advantages of databases and the Database Management System (DBMS), in the context of the electronic bookstore are discussed, followed by the emphasizing of the various business rules that govern the said database, the display of the Entity Relationship Diagram (Crow's Foot Notation) and the Database Diagram, and subsequently, the SQL statements in terms of Data Definition Language (DDL) and Data Manipulation Language (DML).

In the context of the E-Bookstore, having implement a database would greatly increase the efficiency of the bookstore's ability to manage and use data. Not only is the E-Bookstore able to store large amounts of data ranging from the most basics of information such as the particulars of its members all the way to the entire E-Bookstore's extensive inventory and supply-chain record, both in a systematic and organized manner, a database also allows for the process of retrieving data to be done in a much seamless and streamlined manner. Besides storage and the retrieval of data, setting up a database management system for the E-Bookstore would also enable for better management of the bookstore's inventories, more importantly so since the bookstore is an online business model. In addition to that, implementation of a database management system ensures that data consistency within the bookstore's ecosystem is maintained at the highest level, on grounds that there are various data validation rules and constraints which were put in place to govern the said database. At the same time, because a database allows for the sharing of information across all levels of stakeholders of the organization, thereby making it unnecessary to have duplicating data, data redundancy within the E-Bookstore is hence said to be greatly reduced as a result of the implementation of such a database management system.

Improved integration of the E-Bookstore's data would also be possible as the E-Bookstore's database could then be used to integrate with other systems and platforms connected to the bookstore, such as its delivery partners or its supply chain partners. Informative business insights, future prospect of the E-Bookstore's business model and a more personalized experience for its members would then all be beneficial by-products of such improvement in its data integration. Lastly, in the event of a system failure within the E-Bookstore, data management system is able to ensure that no data would be lost in such events because data has already been backed up into the system, essentially allowing for the recovery and restoration of the E-Bookstore's data. In short, comparing traditional file-based systems with

the database management system, the latter provides more benefits over the former specifically in terms of data storage, data retrieval, improved inventory management for the E-Bookstore, data consistency, better data sharing, reduced data redundancy, improved data integration across various systems and platforms, and also improved backup and data recovery mechanisms.

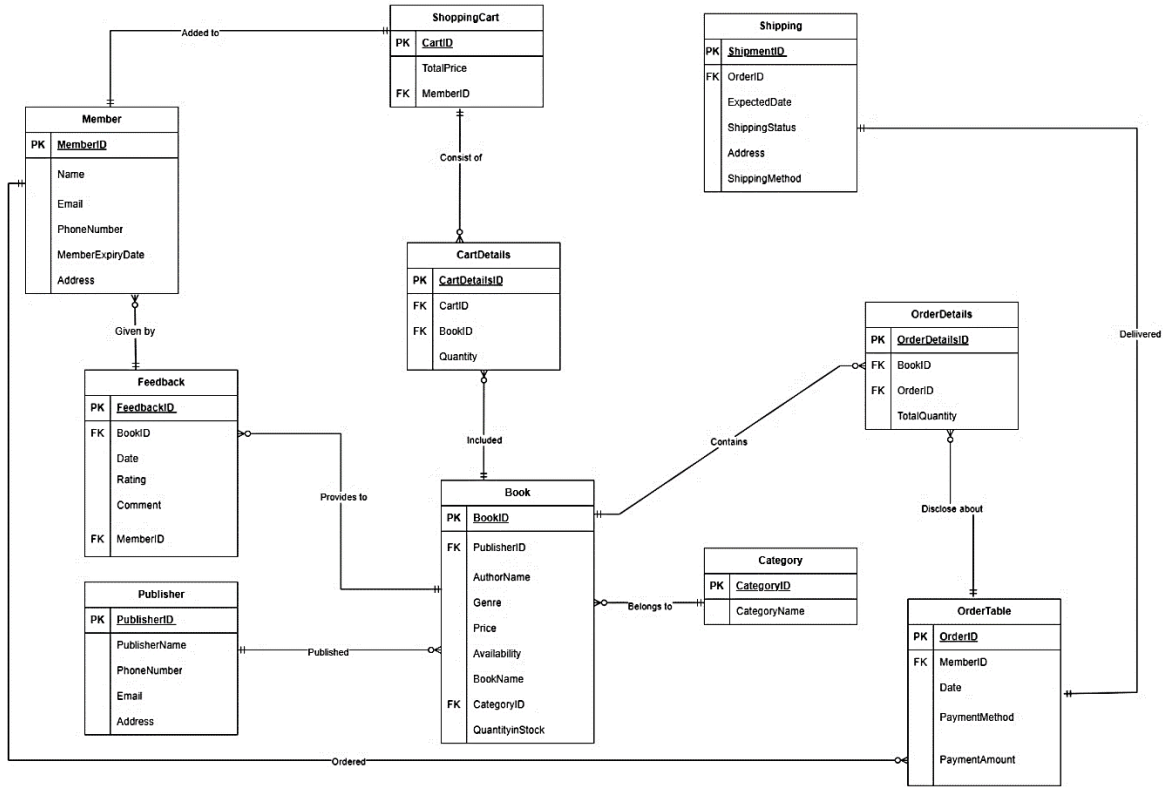
2.0 Business Rules

Below is a list of the business rules that govern the E-Bookstore's database system:

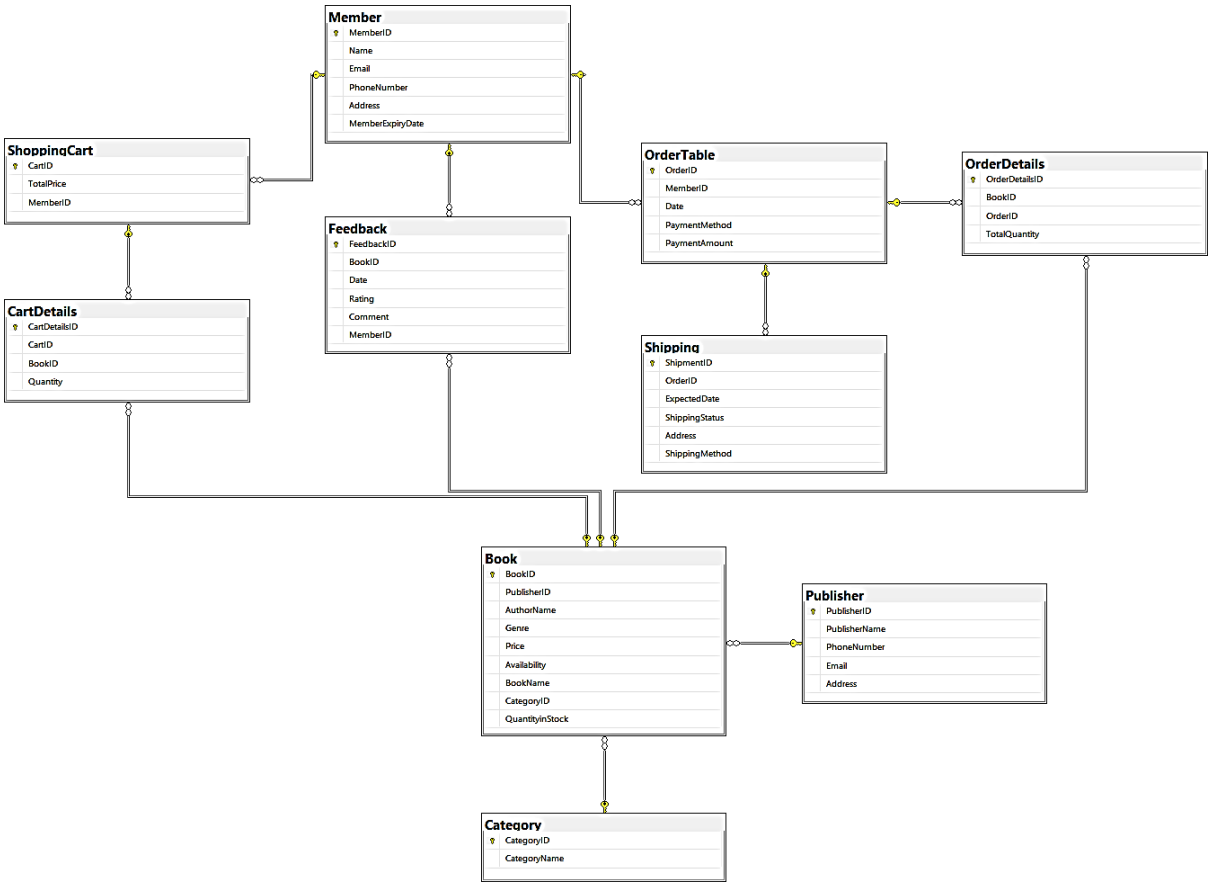
1. One MEMBER can only have one SHOPPINGCART. At the same time, one SHOPPINGCART must belong to one MEMBER only. Hence, there is only one *CartID* per member.
2. Each MEMBER will only have one *MemberID* and the *MemberID* would still remain the same even after the renewal of membership upon expiry.
3. One MEMBER can only give one FEEDBACK per book. FEEDBACK includes a rating from 1 to 10 (1 = Terrible; 10 = Masterpiece) and a short comment which is optional. Note that FEEDBACK cannot be changed once it has been made.
4. One FEEDBACK can only be given to one BOOK but one BOOK could have many FEEDBACKs.
5. One PUBLISHER can publish many BOOKs but each BOOK must belong to only one PUBLISHER.
6. *Price* in the BOOK table represents the individual price of a particular book.
7. One SHOPPINGCART can contain many BOOKs. At the same time, one BOOK can be added to many SHOPPINGCARTs.
8. *Quantity* in CARTDETAILS refers to the total quantity of BOOKs in the SHOPPINGCART.
9. ORDERTABLE represents the order placed by each member.
10. The *PaymentMethod* in ORDERTABLE refers to the total amount payable for a single order.
11. One BOOK can belong to many ORDERTABLE. At the same time, one ORDERTABLE can contain many BOOKs.
12. One MEMBER can have many ORDERTABLE but one ORDERTABLE must belong to one MEMBER only.

13. *TotalQuantity* in ORDERDETAILS refers to the total quantity of books placed within a single order.
14. There is a distinction between SHOPPINGCART and ORDERTABLE because books placed in the SHOPPINGCART need not be purchased but books placed in the ORDERTABLE is definitely going to be purchased.
15. One ORDERTABLE can only belong to one SHIPPING. At the same time, one SHIPPING must belong to one ORDERTABLE only. Hence, there must be one unique *ShipmentID* for each order placed.
16. The *Address* in MEMBER need not be the same as the *Address* in *SHIPPING*. This is because members could decide to send their shipments to another address other than the one registered with the E-Bookstore.
17. The *ExpectedDate* in SHIPPING must be within 7 working days after an order (represented by ORDERTABLE) has been placed. This means that the difference between the *Date* in ORDERTABLE and the *ExpectedDate* in SHIPPING must not exceed 7 working days.
18. One CATEGORY can have many BOOKs but one BOOK can only belong to one CATEGORY.

3.0 Entity Relationship Diagram



4.0 Database Diagram



5.0 SQL Statements

5.1 Data Definition Language (DDL)

```
CREATE DATABASE EBookstore;

USE EBookstore;
```

Input for Member Table:

```
-- Creating the Member Table
CREATE TABLE Member
(MemberID Int Primary Key,
Name nvarchar(50),
Email nvarchar(50),
PhoneNumber varchar(15),
Address nvarchar(50),
MemberExpiryDate Date);

-- Insert values into the Member Table
INSERT INTO Member VALUES
(1, 'Lily Whitney', 'lilyw@gmail.com', '03-6204-5197', '172 Jln Petaling', '2023-08-26'),
(2, 'Lea Lim', 'leal@gmail.com', '03-4149-0972', '11 Jln Merah', '2024-05-16'),
(3, 'Andrew Holts', 'andrewh@gmail.com', '04-5566-0972', '40 Mont Kiara', '2025-01-03'),
(4, 'Simon Peter', 'simonp@gmail.com', '03-1257-9335', '65 Jln Kangkok', '2026-04-20'),
(5, 'Thea Middleton', 'theam@gmail.com', '05-6734-2793', '8 Jln Kuning', '2023-12-12'),
(6, 'Noel Chan', 'noelc@gmail.com', '04-1223-5643', '2 Jln Hijau', '2024-06-16'),
(7, 'Britney Williams', 'britneyw@gmail.com', '05-3352-6780', '56 Jln Stampin', '2027-05-28'),
(8, 'Thaddeus Hammington', 'thaddeush@gmail.com', '04-5611-9876', '19 Jln Biru', '2030-08-12'),
(9, 'Theodore Page', 'theordorep@gmail.com', '03-5514-0002', '89 Jln Ong', '2023-12-29'),
(10, 'Tiffany Miller', 'tiffanym@gmail.com', '05-6661-1223', '66 Jln Stutong', '2029-07-01');
```

Output for Member Table:

Results | Messages

	MemberID	Name	Email	PhoneNumber	Address	MemberExpiryDate
1	1	Lily Whitney	lilyw@gmail.com	03-6204-5197	172 Jln Petaling	2023-08-26
2	2	Lea Lim	leal@gmail.com	03-4149-0972	11 Jln Merah	2024-05-16
3	3	Andrew Holts	andrewh@gmail.com	04-5566-0972	40 Mont Kiara	2025-01-03
4	4	Simon Peter	simonp@gmail.com	03-1257-9335	65 Jln Kangkok	2026-04-20
5	5	Thea Middleton	theam@gmail.com	05-6734-2793	8 Jln Kuning	2023-12-12
6	6	Noel Chan	noelc@gmail.com	04-1223-5643	2 Jln Hijau	2024-06-16
7	7	Britney Williams	britneyw@gmail.com	05-3352-6780	56 Jln Stampin	2027-05-28
8	8	Thaddeus Hammington	thaddeush@gmail.com	04-5611-9876	19 Jln Biru	2030-08-12
9	9	Theodore Page	theordorep@gmail.com	03-5514-0002	89 Jln Ong	2023-12-29
10	10	Tiffany Miller	tiffanym@gmail.com	05-6661-1223	66 Jln Stutong	2029-07-01

Input for ShoppingCart Table:

```
-- Creating the ShoppingCart Table
CREATE TABLE ShoppingCart
(CartID Int Primary Key,
TotalPrice decimal(10,2),
MemberID INT,
Foreign Key (MemberID) References Member(MemberID));

-- Insert values into the ShoppingCart Table
INSERT INTO ShoppingCart VALUES
(1, 54.55, 1),
(2, 20.60, 2),
(3, 45.05, 3),
(4, 30.99, 4),
(5, 70.65, 5),
(6, 42.80, 6),
(7, 15.90, 7),
(8, 88.94, 8),
(9, 61.03, 9),
(10, 33.95, 10);
```

Output for ShoppingCart Table:

Results | Messages

	CartID	TotalPrice	MemberID
1	1	54.55	1
2	2	20.60	2
3	3	45.05	3
4	4	30.99	4
5	5	70.65	5
6	6	42.80	6
7	7	15.90	7
8	8	88.94	8
9	9	61.03	9
10	10	33.95	10

Input for Publisher Table:

```
-- Creating the Publisher Table
CREATE TABLE Publisher
(PublisherID Int Primary Key,
PublisherName nvarchar(50),
PhoneNumber varchar(15),
Email nvarchar(50),
Address nvarchar(50));

-- Insert values into the Publisher Table
INSERT INTO Publisher VALUES
(1, 'SAGE Publishing', '07-386 3845', 'sage@gmail.com', '53 Taman Bukit'),
(2, 'Pearson', '05-658 8464', 'pearson@gmail.com', '42 Bukit Mertajam'),
(3, 'Abrams Books', '016-895 9498 ', 'abramsb@gmail.com', '20 Pending'),
(4, 'Wiley', '06-462 2543', 'wiley@gmail.com', '5 Bau'),
(5, 'Springer', '012-748 0744', 'springer@gmail.com', '22 Betong Street'),
(6, 'Scholastic', '05-878 0815 ', 'scholastic@gmail.com', '11 Taman Tun Jugah'),
(7, 'Chronicle Books', '05-285 8981', 'chronicleb@gmail.com', '76 Ahmad Zaid'),
(8, 'Penguin', '05-624 1244 ', 'penguin@gmail.com', '15 Matang Jaya'),
(9, 'Holiday House', '06-652 1641', 'holidayh@gmail.com', '41 Long Luteng'),
(10, 'Workman Publishing', '09-321 1324 ', 'workmanp@gmail.com', '1 Kuala Lipis');
```

Output for Publisher Table:

Results Messages

	PublisherID	PublisherName	PhoneNumber	Email	Address
1	1	SAGE Publishing	07-386 3845	sage@gmail.com	53 Taman Bukit
2	2	Pearson	05-658 8464	pearson@gmail.com	42 Bukit Mertajam
3	3	Abrams Books	016-895 9498	abramsb@gmail.com	20 Pending
4	4	Wiley	06-462 2543	wiley@gmail.com	5 Bau
5	5	Springer	012-748 0744	springer@gmail.com	22 Betong Street
6	6	Scholastic	05-878 0815	scholastic@gmail.com	11 Taman Tun Jugah
7	7	Chronicle Books	05-285 8981	chronicleb@gmail.com	76 Ahmad Zaid
8	8	Penguin	05-624 1244	penguin@gmail.com	15 Matang Jaya
9	9	Holiday House	06-652 1641	holidayh@gmail.com	41 Long Luteng
10	10	Workman Publishing	09-321 1324	workmanp@gmail.com	1 Kuala Lipis

Input for Category Table:

```
--Creating the Category Table
CREATE TABLE Category
(CategoryID Int Primary Key,
CategoryName nvarchar(50));

--Insert values into the Category Table
INSERT INTO Category VALUES
(1, 'Loan for 1 day'),
(2, 'Loan for 1 weeks'),
(3, 'Loan for 2 weeks'),
(4, 'Loan for 3 weeks'),
(5, 'Loan for 1 month');
```

Output for Category Table:

Results | Messages

	CategoryID	CategoryName
1	1	Loan for 1 day
2	2	Loan for 1 weeks
3	3	Loan for 2 weeks
4	4	Loan for 3 weeks
5	5	Loan for 1 month

Input for Book Table:

```
-- Creating the Book Table
CREATE TABLE Book
(BookID Int Primary Key,
PublisherID Int, Foreign Key (PublisherID) References Publisher(PublisherID),
AuthorName nvarchar(50),
Genre nvarchar(50),
Price decimal(10,2),
Availability nvarchar(50),
BookName nvarchar(50),
CategoryID Int, Foreign Key (CategoryID) References Category(CategoryID),
QuantityinStock Int);

-- Insert values into the Book Table
INSERT INTO Book VALUES
(1, 1, 'Jordan B. Peterson', 'Self Help', 69.90, 'Yes', '12 Rules for Life',1,10),
(2, 2, 'Toni Morrison', 'Romance', 45.50, 'Yes', 'Beloved',2,20),
(3, 3, 'Jane Austen', 'Fiction', 79.90, 'Yes', 'Anna Karenina',3,40),
(4, 4, 'Ray Bradbury', 'Novel', 34.90, 'Yes', 'Fahrenheit 451',4,30),
(5, 5, 'Alice Walker', 'Science Fiction', 80.15, 'Yes', 'The Color Purple',5,35),
(6, 6, 'Harper Lee', 'Horror', 40.90, 'Yes', 'To Kill a Mockingbird',3,20),
(7, 7, 'J. R. R. Tolkien', 'Young Adult Fiction', 53.90, 'Yes', 'The Hobbit',4,15),
(8, 8, 'Haruki Murakami', 'Novel', 58.70, 'Yes', 'Kafka On The Shore',3,20),
(9, 9, 'Ernest Hemingway', 'Fiction', 35.90, 'Yes', 'The Old Man and the Sea',2,16),
(10, 10, 'Mary Shelley', 'Thriller', 87.90, 'Yes', 'Frankenstein',1,34);
```

Output for Book Table:

Results Messages

	BookID	PublisherID	AuthorName	Genre	Price	Availability	BookName	CategoryID	QuantityinStock
1	1	1	Jordan B. Peterson	Self Help	69.90	Yes	12 Rules for Life	1	10
2	2	2	Toni Morrison	Romance	45.50	Yes	Beloved	2	20
3	3	3	Jane Austen	Fiction	79.90	Yes	Anna Karenina	3	40
4	4	4	Ray Bradbury	Novel	34.90	Yes	Fahrenheit 451	4	30
5	5	5	Alice Walker	Science Fiction	80.15	Yes	The Color Purple	5	35
6	6	6	Harper Lee	Horror	40.90	Yes	To Kill a Mockingbird	3	20
7	7	7	J. R. R. Tolkien	Young Adult Fiction	53.90	Yes	The Hobbit	4	15
8	8	8	Haruki Murakami	Novel	58.70	Yes	Kafka On The Shore	3	20
9	9	9	Ernest Hemingway	Fiction	35.90	Yes	The Old Man and the Sea	2	16
10	10	10	Mary Shelley	Thriller	87.90	Yes	Frankenstein	1	34

Input for Feedback Table:

```
-- Creating the Feedback Table
CREATE TABLE Feedback
(FeedbackID Int Primary Key,
BookID Int, Foreign Key (BookID) References Book(BookID),
Date Date,
Rating Int,
Comment nvarchar(50),
MemberID Int, Foreign Key (MemberID) References Member(MemberID));

-- Insert values into the Feedback Table
INSERT INTO Feedback VALUES
(1, 1, '2023-01-01', 10, 'Interesting book', 1),
(2, 2, '2023-02-05', 10, 'What a masterpiece', 2),
(3, 3, '2022-10-29', 9, 'The plot was great', 3),
(4, 4, '2020-11-03', 1, 'What a waste of money', 4),
(5, 5, '2021-03-06', 2, 'Definetely not recommending', 5),
(6, 6, '2020-04-01', 7, 'Favorite book so far', 6),
(7, 7, '2023-06-16', 4, 'It was okay only', 7),
(8, 8, '2023-01-18', 8, 'A good read indeed', 8),
(9, 9, '2020-10-19', 3, 'Not something I would read again', 9),
(10, 10, '2022-08-30', 10, 'It was just genius', 10);
```

Output for Feedback Table:

Results Messages

	FeedbackID	BookID	Date	Rating	Comment	MemberID
1	1	1	2023-01-01	10	Interesting book	1
2	2	2	2023-02-05	10	What a masterpiece	2
3	3	3	2022-10-29	9	The plot was great	3
4	4	4	2020-11-03	1	What a waste of money	4
5	5	5	2021-03-06	2	Definetely not recommending	5
6	6	6	2020-04-01	7	Favorite book so far	6
7	7	7	2023-06-16	4	It was okay only	7
8	8	8	2023-01-18	8	A good read indeed	8
9	9	9	2020-10-19	3	Not something I would read again	9
10	10	10	2022-08-30	10	It was just genius	10

Input for CartDetails Table:

```
-- Creating the CartDetails Table
CREATE TABLE CartDetails
(CartDetailsID Int Primary Key,
CartID Int, Foreign Key (CartID) References ShoppingCart(CartID),
BookID Int, Foreign Key (BookID) References Book(BookID),
Quantity Int);

-- Insert values into the CartDetails Table
INSERT INTO CartDetails VALUES
(1, 1, 1, 5),
(2, 2, 2, 1),
(3, 3, 3, 3),
(4, 4, 4, 5),
(5, 5, 5, 4),
(6, 6, 6, 8),
(7, 7, 7, 3),
(8, 8, 8, 1),
(9, 9, 9, 6),
(10, 10, 10, 7);
```

Output for CartDetails Table:

Results | Messages

	CartDetailsID	CartID	BookID	Quantity
1	1	1	1	5
2	2	2	2	1
3	3	3	3	3
4	4	4	4	5
5	5	5	5	4
6	6	6	6	8
7	7	7	7	3
8	8	8	8	1
9	9	9	9	6
10	10	10	10	7

Input for OrderTable Table:

```
-- Creating the OrderTable Table
CREATE TABLE OrderTable
(OrderID Int Primary Key,
MemberID Int, Foreign Key (MemberID) References Member(MemberID),
Date Date,
PaymentMethod nvarchar(50),
PaymentAmount decimal(10,2));

-- Insert values into the OrderTable Table
INSERT INTO OrderTable VALUES
(1, 1, '2023-06-16', 'Credit Card', 100.90),
(2, 2, '2023-06-01', 'Credit Card', 99.80),
(3, 3, '2023-05-01', 'Credit Card', 55.60),
(4, 4, '2023-07-01', 'Internet Banking', 44.90),
(5, 5, '2023-07-11', 'Credit Card', 120.90),
(6, 6, '2023-05-29', 'Credit Card', 150.90),
(7, 7, '2023-06-12', 'Internet Banking', 70.50),
(8, 8, '2023-07-02', 'Credit Card', 44.30),
(9, 9, '2023-07-14', 'Credit Card', 65.40),
(10, 10, '2023-07-09', 'Credit Card', 60.30);
```

Output for OrderTable Table:

Results Messages

	OrderID	MemberID	Date	PaymentMethod	PaymentAmount
1	1	1	2023-06-16	Credit Card	100.90
2	2	2	2023-06-01	Credit Card	99.80
3	3	3	2023-05-01	Credit Card	55.60
4	4	4	2023-07-01	Internet Banking	44.90
5	5	5	2023-07-11	Credit Card	120.90
6	6	6	2023-05-29	Credit Card	150.90
7	7	7	2023-06-12	Internet Banking	70.50
8	8	8	2023-07-02	Credit Card	44.30
9	9	9	2023-07-14	Credit Card	65.40
10	10	10	2023-07-09	Credit Card	60.30

Input for OrderDetails Table:

```
-- Creating the OrderDetails Table
CREATE TABLE OrderDetails
(OrderDetailsID Int Primary Key,
BookID Int, Foreign Key (BookID) References Book(BookID),
OrderID Int, Foreign Key (OrderID) References OrderTable(OrderID),
TotalQuantity Int);

-- Insert values into the OrderDetails Table
INSERT INTO OrderDetails VALUES
(1, 1, 1, 2),
(2, 2, 2, 3),
(3, 3, 3, 1),
(4, 4, 4, 2),
(5, 5, 5, 4),
(6, 6, 6, 3),
(7, 7, 7, 6),
(8, 8, 8, 5),
(9, 9, 9, 2),
(10, 10, 10, 1);
```

Output for OrderDetails Table:

Results | Messages

	OrderDetailsID	BookID	OrderID	TotalQuantity
1	1	1	1	2
2	2	2	2	3
3	3	3	3	1
4	4	4	4	2
5	5	5	5	4
6	6	6	6	3
7	7	7	7	6
8	8	8	8	5
9	9	9	9	2
10	10	10	10	1

Input for Shipping Table:

```
-- Creating the Shipping Table
CREATE TABLE Shipping
(ShipmentID Int Primary Key,
OrderID Int, Foreign Key (OrderID) References OrderTable(OrderID),
ExpectedDate Date,
ShippingStatus nvarchar(50),
Address nvarchar(50),
ShippingMethod nvarchar(50));

-- Insert values into the Shipping Table
INSERT INTO Shipping VALUES
(1, 1, '2023-06-26', 'Yes', '1 Bukit Gajah', 'Freight'),
(2, 2, '2023-06-09', 'Yes', '2 Batu Selabat', 'Freight'),
(3, 3, '2023-05-09', 'Yes', '45 Gunung Timah', 'Sea'),
(4, 4, '2023-07-11', 'Yes', '7 Demak Laut', 'Freight'),
(5, 5, '2023-07-19', 'No', '13 Samarahan', 'Freight'),
(6, 6, '2023-06-06', 'Yes', '87 Jaaln Sejingkat', 'Sea'),
(7, 7, '2023-06-20', 'Yes', '83 Rock Road', 'Freight'),
(8, 8, '2023-07-11', 'Yes', '5 Chester Park', 'Freight'),
(9, 9, '2023-07-24', 'No', '7 Tabuan Jaya', 'Freight'),
(10, 10, '2023-07-18', 'No', '9 Padungan Road', 'Sea');
```

Output for Shipping Table:

Results Messages

	ShipmentID	OrderID	ExpectedDate	ShippingStatus	Address	ShippingMethod
1	1	1	2023-06-26	Yes	1 Bukit Gajah	Freight
2	2	2	2023-06-09	Yes	2 Batu Selabat	Freight
3	3	3	2023-05-09	Yes	45 Gunung Timah	Sea
4	4	4	2023-07-11	Yes	7 Demak Laut	Freight
5	5	5	2023-07-19	No	13 Samarahan	Freight
6	6	6	2023-06-06	Yes	87 Jaaln Sejingkat	Sea
7	7	7	2023-06-20	Yes	83 Rock Road	Freight
8	8	8	2023-07-11	Yes	5 Chester Park	Freight
9	9	9	2023-07-24	No	7 Tabuan Jaya	Freight
10	10	10	2023-07-18	No	9 Padungan Road	Sea

5.2 Data Manipulation Language (DML)

5.2.1 Question 2 (b) (i)

Input:

```
--Question 2 b (i)
SELECT
    Book.BookID,
    Book.BookName AS BookTitle,
    COUNT(Feedback.FeedbackID) AS TotalFeedbacksPerBook
FROM
    Feedback
INNER JOIN
    Book ON Book.BookID = Feedback.BookID
GROUP BY
    Book.BookID, Book.BookName;
```

Output:

Results | Messages

	BookID	BookTitle	TotalFeedbacksPerBook
1	1	12 Rules for Life	1
2	2	Beloved	1
3	3	Anna Karenina	1
4	4	Fahrenheit 451	1
5	5	The Color Purple	1
6	6	To Kill a Mockingbird	1
7	7	The Hobbit	1
8	8	Kafka On The Shore	1
9	9	The Old Man and the Sea	1
10	10	Frankenstein	1

5.2.2 Question 2 (b) (ii)

Input:

```
--Question 2 b (ii)
SELECT
    Member.MemberID,
    Member.Name AS MemberName,
    COUNT(Feedback.FeedbackID) AS TotalFeedbacksPerMember
FROM
    Feedback
LEFT JOIN
    Member ON Member.MemberID = Feedback.MemberID
GROUP BY
    Member.MemberID, Member.Name;
```

Output:

Results Messages

	MemberID	MemberName	TotalFeedbacksPerMember
1	1	Lily Whitney	1
2	2	Lea Lim	1
3	3	Andrew Holts	1
4	4	Simon Peter	1
5	5	Thea Middleton	1
6	6	Noel Chan	1
7	7	Britney Williams	1
8	8	Thaddeus Hammington	1
9	9	Theodore Page	1
10	10	Tiffany Miller	1

5.2.3 Question 2 (b) (iii)

Input:

```
--Question 2 b (iii)
-- To show the total number of books published by each publisher.
SELECT
    Publisher.PublisherID,
    Publisher.PublisherName,
    COUNT(Book.BookID) AS TotalNumberOfBooksPublished
FROM
    Publisher
Left JOIN
    Book ON Publisher.PublisherID = Book.PublisherID
GROUP BY
    Publisher.PublisherID, Publisher.PublisherName;
```

Output:

Results Messages

	PublisherID	PublisherName	TotalNumberOfBooksPublished
1	1	SAGE Publishing	1
2	2	Pearson	1
3	3	Abrams Books	1
4	4	Wiley	1
5	5	Springer	1
6	6	Scholastic	1
7	7	Chronicle Books	1
8	8	Penguin	1
9	9	Holiday House	1
10	10	Workman Publishing	1

5.2.4 Question 2 (b) (iv)

Input:

```
--Question 2 (b) (iv)
SELECT
    Category.CategoryID,
    Category.CategoryName,
    SUM(Book.QuantityinStock) AS TotalNumberofBooksPerCategory
FROM
    Category
LEFT JOIN
    Book ON Category.CategoryID = Book.CategoryID
GROUP BY
    Category.CategoryID, Category.CategoryName;
```

Output:

	CategoryID	CategoryName	TotalNumberofBooksPerCategory
1	1	Loan for 1 day	44
2	2	Loan for 1 weeks	36
3	3	Loan for 2 weeks	80
4	4	Loan for 3 weeks	45
5	5	Loan for 1 month	35

5.2.5 Question 2 (b) (v)

Input:

```
--Question 2 b (v)
Select
    BookName
From
    Book
    Where Price > (select avg(GenreQuantity) from Book);
```

Output:

Results | Messages

	BookName
1	12 Rules for Life
2	Anna Karenina
3	The Color Purple
4	Frankenstein

5.2.6 Question 2 (b) (vi)

Input:

```
--Question 2 b (vi)
SELECT
    Book.Genre,
    SUM(QuantityinStock) AS TotalBooks
FROM
    Book
Group By Genre;
```

Output:

Results | Messages

	Genre	TotalBooks
1	Fiction	56
2	Horror	20
3	Novel	50
4	Romance	20
5	Science Fiction	35
6	Self Help	10
7	Thriller	34
8	Young Adult Fiction	15

5.2.7 Question 2 (b) (vii)

Input:

```
--Question 2 b (vii)
SELECT *
FROM
    Member m
WHERE NOT EXISTS
(SELECT
    MemberID
FROM
    OrderTable o); -- Output shows that there are no members who did not make any orders.
```

Output:

Results | Messages

MemberID	Name	Email	PhoneNumber	Address	MemberExpiryDate

5.2.8 Question 2 (b) (viii)

Input:

```
-- Question 2 b (viii)
SELECT
    AVG(Rating) AS AverageRatingPerBook
FROM
    Feedback; -- The average rating for each book is 6.
```

Output:

Results | Messages

	AverageRatingPerBook
1	6

5.2.9 Question 2 (b) (ix)

Input:

```
-- Question 2 b (ix)
SELECT
    SUM(CartDetails.Quantity) AS TotalBooksAddedtoCart
FROM
     CartDetails
RIGHT JOIN
    ShoppingCart ON ShoppingCart.CartID = CartDetails.CartID; --There are in toal 43 books which were added to the shopping cart.
```

Output:

Results | Messages

	TotalBooksAddedtoCart
1	43

5.2.10 Question 2 (b) (x)

Input:

```
--Question 2 b (x)
SELECT
    Member.Name AS MemberName,
    Member.MemberID,
    COUNT(*)
FROM
    Member
INNER JOIN OrderTable
    ON Member.MemberID = OrderTable.MemberID
INNER JOIN OrderDetails
    ON OrderTable.OrderID = OrderDetails.OrderID
GROUP BY Member.Name, Member.MemberID
HAVING COUNT(*) > 1; -- The output shows that no members have made more than 2 orders.
```

Output:

Results | Messages

	MemberName	MemberID	(No column name)