Module Code: CT087-0-M

Module Name: Database for Data Science

Student's TP: TP072295

Student's Name: Afif Rifa'ie bin Mohd Gazali

Lecturer's Name: Mr. Lee Kim Keong

Table of content

1. Introduction

APU E-Bookstore is a book seller located in Kuala Lumpur Malaysia. It has vast collection of books and keep the books in its warehouse. The bookstore ordered from multiple sources of vendor and it sell the book to the customers. As the business grow, the bookstore wants to implement an online store to facilitate the purchase of books to more customers. This report is to explore the benefit of adopting online database, to design and implement a database management system for the bookstore. The first part of this report begins by discussing the advantages of a database and DBMS, followed by an exploration of business rules. Furthermore, it involves creating an Entity Relationship Diagram using crow's foot notation with the Ms. Visio tool. Lastly, the report includes the creation of a data diagram and the generation of SQL statements to explain Data Definition Language (DDL) and Data Manipulation Language (DML).

2. The advantage of DBMS are as follows;

Simple Model

A DBMS does not require complex and in structuring and querying. As the structure operation is simple and straight forward, it is sufficient to be handled with simple SQP quarries such as insertion, deletion or creation of file or data

and it is cost effective as the bookstore does not require an expertise to operate the database.

Data Integrity

A DBMS makes sure that the data in a database is consistent, synchronize and reliable. The DBMS provides constraints and consistency checks mechanism which trigger an error when any violation in the SQL statement. This is necessary for the store to keep correct records of books, customers, sales, and reviews and to avoid human error when operating the database.

Data Security

The APUebookStore have access to sensitive information of its customers such as customer's name, contact number, address and financial transaction. DBMS security features protect the information from unauthorized access. The GRANT statement in SQL, give specific access to specific individuals and limit the user to perform any unnecessary adjustment to the database.

Data Sharing and Scalability

The DBMS features data-centric which the data is independent to any applications. This allows data to be used in multiple platform and support access by multiple users. This may improve accessibility of the data information and gain business productivity.

Data Independence

A file base data system is significantly dependent on the data. It is important that the application know the structure in order to accessing the data. Any adjustment on the structure and method of the file, it requires modification of the application as well. In contrast, DBMS allows modification of the application without reflecting the structure and access method. The system can be adjusted and improved without disturbing the existing applications.

Automation of Business Processes

A DBMS can handle many business processes, such as managing reviews, order processing, and billing in centralized location. This could help the shop streamline its work and get better and faster results.

Data Accuracy

Data consistency is the state where the data is accurate and not outdated. It is important for the business to ensure accuracy and reliability of the data. DBMS has set of rules and the set of rules can be set to ensure all the data in the APU E-Bookstore database are correct and accepted can be accomplished by the database developer.

Better access to and retrieval of data

A DBMS makes it easy and quick to get to the data. This helps the APU E-Bookstore quickly find information about books, customers, and sales, which is very important for an online store.

Data Recovery

A DBMS has ways to back up and get back data. This is important for the APU E-Bookstore to be able to get back data if the system fails or the data gets damaged. The main recovery features of DBMS are logical recovery, which restoring the data from log file in the database, and physical recovery, involves restoring and repairing damaged or corrupter files.

Concurrent Access

A DBMS allow access more than one person use the database at the same time. This is necessary for the APU E-Bookstore to let many people look at books and buy them online at the same time with real time updates without crashing the system.

Overall, a DBMS can give the APU E-Bookstore a strong and safe way to handle its data and support its online store, which is very important for its expansion and success.

3. Business Rules

Business Rules is company's operational policies, processes, and logical set of rules. Business rules are the guiding principles for how a company handles its data. It acts as a plan to make sure that data stays accurate and useful throughout its lifecycle in the APUebookstore.

Purchasing and Inventory Management Rules.

1. Book Ordering:
    a. The bookstore manager compiles a list of required books and places orders with publishers.
2. Book Supply:
    a. Publishers supply the ordered books to the bookstore.
3. Inventory Update:
    a. The bookstore manager records details of newly arrived books in the bookstore's inventory.
4. Invoice Processing:
    a. An invoice is generated for each order and sent to the accounts department for processing and payment.

Online Purchase and Membership Rules.

5. Membership Requirement:
    a. Individuals must register as members to make online purchases.

6. Book Viewing:
    a. Members can view book information and read reviews without making a purchase.
7. Book Selection:
    a. Members can select books into the website’s shopping cart.
8. Shopping Cart Summary:
    a. The shopping cart provides a summary of the selected books and the total cost.
9. Payment and Delivery:
    a. Purchases are completed once payment is made.
    b. The bookstore sends the purchased books to members within 7 working days.

Database Management Rules.

10. Book Information Management:
    a. The database manages information about books available in the bookstore.
11. Member Information Management:
    a. The database stores information about members and the books they have ordered.
12. Review Submission Rules:
    a. Review Submission: Members can provide reviews for books, including a rating on a scale of 1 to 10.

b. The rating scale is defined as 1 (terrible) to 10 (masterpiece).

c. Each member is allowed to submit only one review per book.

13. Review Integrity.

a. No changes are allowed to submitted reviews.

4. Entity Relationship Diagram (ERD Crow's Foot Notation)

To illustrate the relationship between the entities in the APU E-Bookstore database, the Entity Relationship Diagram is widely used with Crow's Foot Notation. The entities in the dataset are connected by lines and symbols at the end of each table to describe the cardinality of the relationship between the entities. Below is the ERD to describe the entities in the APU E-Bookstore.

Member-to-Feedback Relationship (One-to-Many):
One Member can provide feedback on Many Books.
Cardinality: Member (1) ---< Feedback (Many)

Publisher-to-Book Relationship (One-to-Many):
One Publisher can publish Many Books.
Cardinality: Publisher (1) ---< Book (Many)

Book-to-Feedback Relationship (One-to-Many):
One Book can receive feedback from Many Members.
Cardinality: Book (1) ---< Feedback (Many)

Manager-to-BookOrder Relationship (One-to-Many):

One Manager can place Many Book Orders.

Cardinality: Manager (1) ---< BookOrder (Many)

Member-to-BookOrder Relationship (One-to-Many):

One Member can place Many Book Orders.

Cardinality: Member (1) ---< BookOrder (Many)

Book-to-BookOrder Relationship (One-to-Many):

One Book can be part of Many Book Orders.

Cardinality: Book (1) ---< BookOrder (Many)

BookOrder-to-DeliveryStatus Relationship (One-to-One):

One Book Order can have One Delivery Status.

Cardinality: BookOrder (1) --- DeliveryStatus (1)

(the remainder of this page is intentionally left blank)

4.a. Entity Relationship Diagram using crow's foot notation

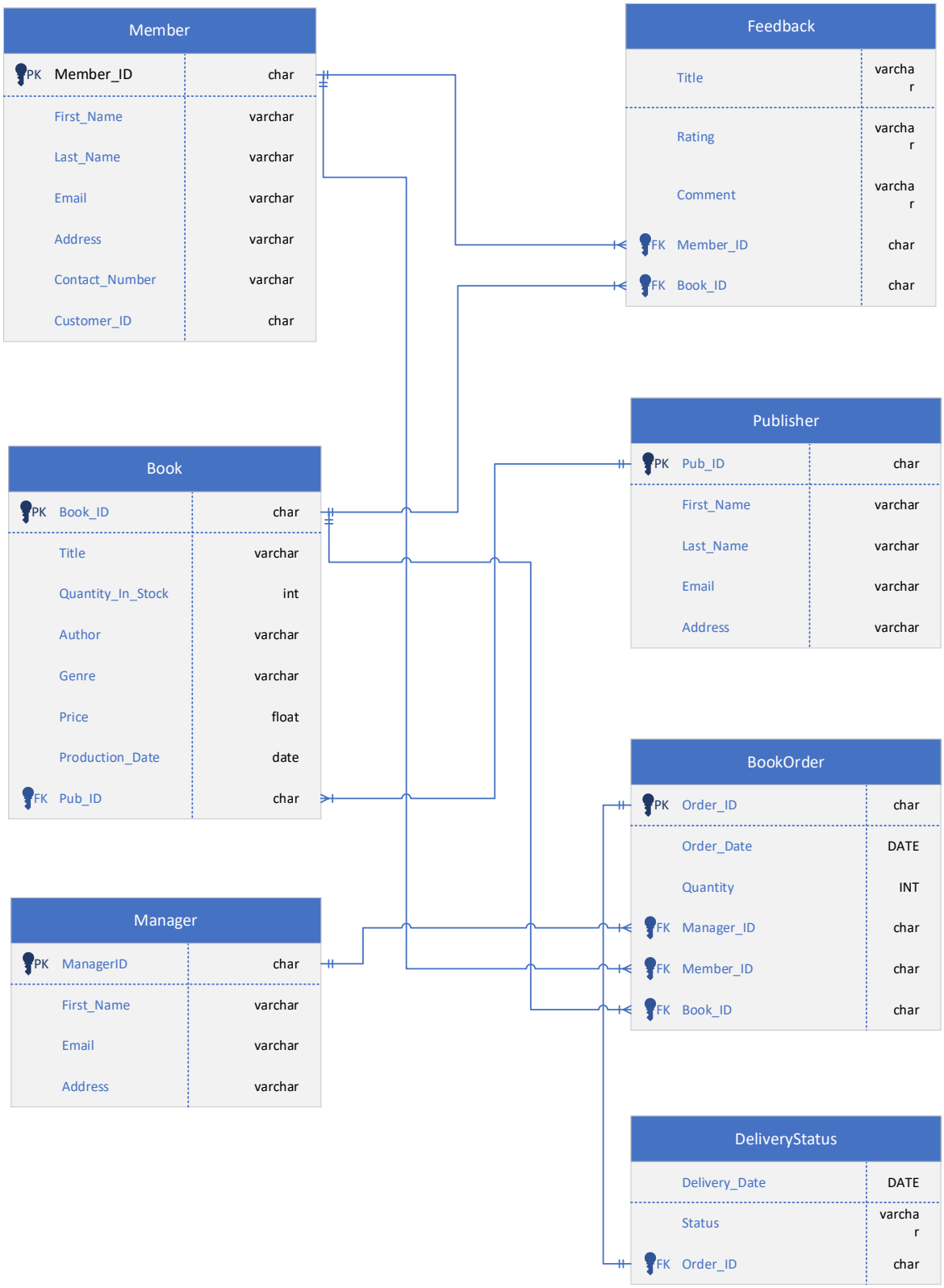

Illustration 1 Entity Relationship Diagram (Crow's Foot Notation)

5. Database schema Diagram – Database diagram reverse engineering

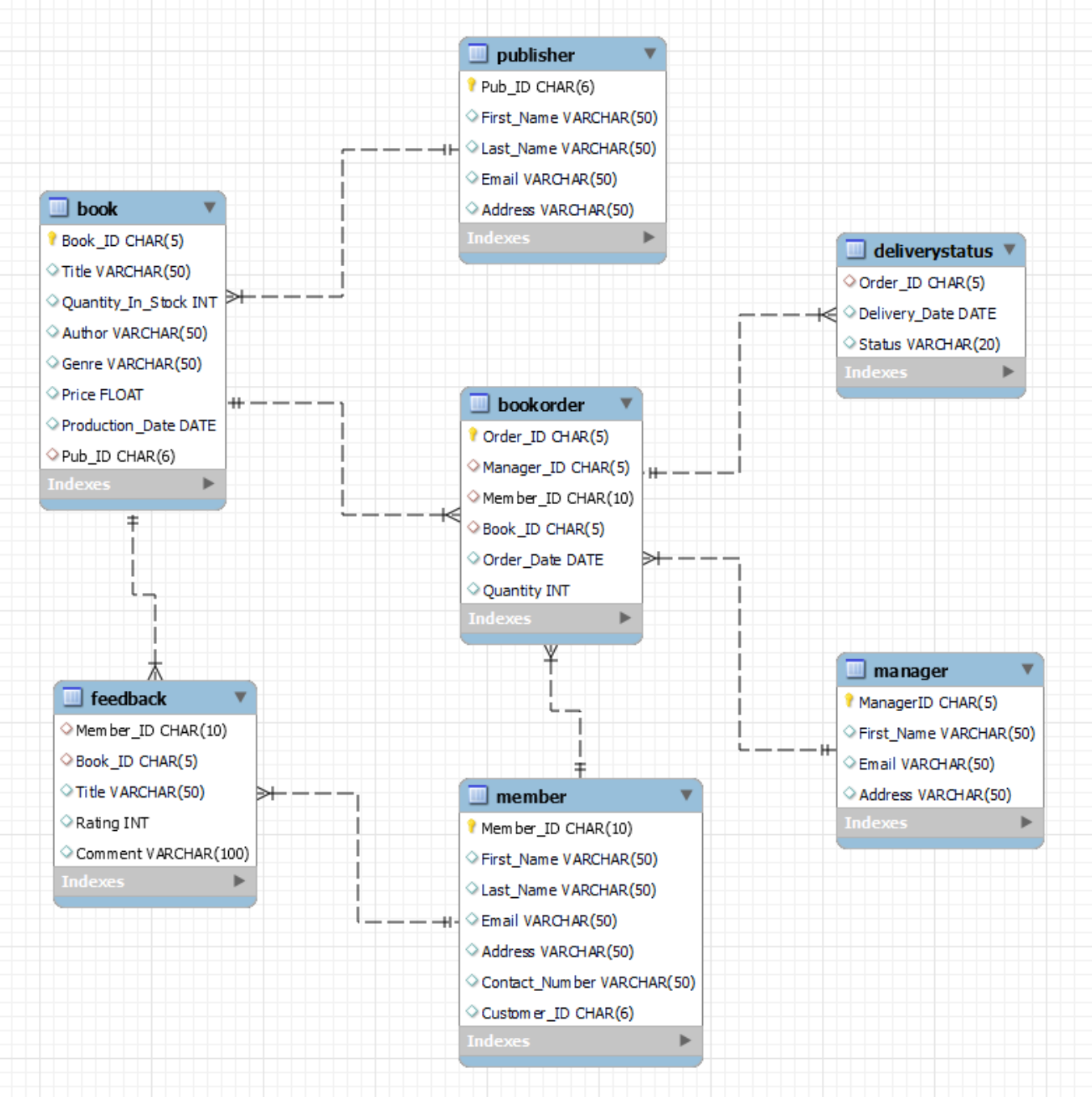

Illustration 2 Database schema – Database diagram reverse engineering

6. SQL – Data Definition Language (DDL)

CREATE DATABASE apuebookstore;:

USE apuebookstore;:

CREATE TABLE Member(...);:

```sql
CREATE DATABASE apuebookstore; -- creates a new database named 'apuebookstore'

USE apuebookstore; -- specifies that subsequent SQL statements will be executed in the 'apuebookstore' database

-- creates a table named Member
-- table has columns for Member_ID, First_Name, Last_Name, Email, Address, Contact_Number, and Customer_ID.
-- The Member_ID column is designated as the primary key.

CREATE TABLE Member(
    Member_ID char(10) primary key,
    First_Name varchar(50),
    Last_Name varchar(50),
    Email varchar(50),
    Address varchar(50),
    Contact_Number varchar(50),
    Customer_ID char(6)
    );

-- command inserts data into the 'Member' table.
INSERT INTO apuebookstore.Member VALUES
    ('MBR001', 'Khan', 'Azmeer', 'azmeer@gmail.com', 'Ampang', '012-9264839','CST001'),
    ('MBR002', 'Aisha', 'Lee', 'aisha.lee@yahoo.com', 'Petaling Jaya', '011-3456789','CST002'),
    ('MBR003', 'John', 'Doe', 'john.doe@gmail.com', 'Kuala Lumpur', '010-9876543','CST003'),
    ('MBR004', 'Siti', 'Rahman', 'siti.rahman@hotmail.com', 'Shah Alam', '019-8765432','CST004'),
    ('MBR005', 'David', 'Ng', 'david.ng@gmail.com', 'Subang Jaya', '018-7654321','CST005'),
    ('MBR006', 'Kogee', 'Vathi', 'kogee.vathi@yahoo.com', 'Damansara', '017-6543210','CST006'),
    ('MBR007', 'Ahmad', 'Khan', 'ahmad.khan@gmail.com', 'Klang', '016-5432109','CST007'),
    ('MBR008', 'Nina', 'Lim', 'nina.lim@gmail.com', 'Puchong', '015-4321098','CST008'),
    ('MBR009', 'Tina', 'Rajah', 'tinarajah@gmail.com', 'Cheras', '014-3210987','CST009'),
    ('MBR010', 'Linda', 'Chin', 'linda.chin@gmail.com', 'Kepong', '013-2109876','CST010');
```

The output

Member_ID	First_Name	Last_Name	Email	Address	Contact_Number	Customer_ID
MBR001	Khan	Azmeer	azmeer@gmail.com	Ampang	012-9264839	CST001
MBR002	Aisha	Lee	aisha.lee@yahoo.com	Petaling Jaya	011-3456789	CST002
MBR003	John	Doe	john.doe@gmail.com	Kuala Lumpur	010-9876543	CST003
MBR004	Siti	Rahman	siti.rahman@hotmail.com	Shah Alam	019-8765432	CST004
MBR005	David	Ng	david.ng@gmail.com	Subang Jaya	018-7654321	CST005
MBR006	Kogee	Vathi	kogee.vathi@yahoo.com	Damansara	017-6543210	CST006
MBR007	Ahmad	Khan	ahmad.khan@gmail.com	Klang	016-5432109	CST007
MBR008	Nina	Lim	nina.lim@gmail.com	Puchong	015-4321098	CST008
MBR009	Tina	Rajah	tinarajah@gmail.com	Cheras	014-3210987	CST009
MBR010	Linda	Chin	linda.chin@gmail.com	Kepong	013-2109876	CST010
NULL	NULL	NULL	NULL	NULL	NULL	NULL

CREATE TABLE Publisher (...);:

INSERT INTO apuebookstore.publisher VALUES (...);:

```
34      -- creates a table named Publisher
35      -- table has columns for Pub_ID, First_Name, Last_Name, Email, and Address.
36      -- The Pub_ID column is designated as the primary key.
37      CREATE TABLE Publisher (
38      Pub_ID char(6) primary key,
39      First_Name varchar(50),
40      Last_Name varchar(50),
41      Email varchar(50),
42      Address varchar(50)
43      );
44
45      -- command inserts data into the 'Publisher' table.
46      INSERT INTO apuebookstore.publisher VALUES
47          ('PUB001', 'Ong', 'Kiat','kiatong@gmail.com','Bukit Jalil'),
48          ('PUB002', 'Tan', 'Lee','tanlee@gmail.com','Petaling Jaya'),
49          ('PUB003', 'Wan', 'Mohd','wanmohd@gmail.com','Ampang'),
50          ('PUB004', 'Ng', 'Seng','ngseng@hotmail.com','Cheras'),
51          ('PUB005', 'Karisma', 'Thiah', 'karisma@yahoo.com', 'KL Sentral'),
52          ('PUB006', 'Goh', 'Hock','gohhock@gmail.com','Subang Jaya'),
53          ('PUB007', 'Junaida', 'Salleh','junaida.salleh@gmail.com','Shah Alam'),
54          ('PUB008', 'Lee', 'Cheong','leecheong@yahoo.com','Klang'),
55          ('PUB009', 'Amirah', 'Baharudin','amirah.b@gmail.com','Damansara'),
56          ('PUB010', 'Peri', 'Silva','perisilva@gmail.com','Puchong');
```

The output

Pub_ID	First_Name	Last_Name	Email	Address
PUB001	Ong	Kiat	kiatong@gmail.com	Bukit Jalil
PUB002	Tan	Lee	tanlee@gmail.com	Petaling Jaya
PUB003	Wan	Mohd	wanmohd@gmail.com	Ampang
PUB004	Ng	Seng	ngseng@hotmail.com	Cheras
PUB005	Karisma	Thiah	karisma@yahoo.com	KL Sentral
PUB006	Goh	Hock	gohhock@gmail.com	Subang Jaya
PUB007	Junaida	Salleh	junaida.salleh@gmail.com	Shah Alam
PUB008	Lee	Cheong	leecheong@yahoo.com	Klang
PUB009	Amirah	Baharudin	amirah.b@gmail.com	Damansara
PUB010	Peri	Silva	perisilva@gmail.com	Puchong
NULL	NULL	NULL	NULL	NULL

CREATE TABLE Book (...);:

INSERT INTO apuebookstore.book VALUES (...);:

```
60      -- creates a table named Book
61      -- table has columns for Book_ID, Title, Quantity_In_Stock, Author, Genre, Price, Production_Date, and Pub_ID.
62      -- The Book_ID is designated as the primary key.
63      -- The Pub_ID is designated as the foreign key.
64      CREATE TABLE Book(
65      Book_ID char(5) primary key,
66      Title varchar(50),
67      Quantity_In_Stock int,
68      Author varchar(50),
69      Genre varchar(50),
70      Price float,
71      Production_Date date,
72      Pub_ID char(6),
73      FOREIGN KEY (Pub_ID) REFERENCES Publisher(Pub_ID)
74      );
75
76      -- command inserts data into the 'Book' table.
77      INSERT INTO apuebookstore.Book VALUES
78          ('BK001', 'HP Deathly Hallows', 235, 'Rowling', 'fantasy', 59.90, '2010-11-19', 'PUB001'),
79          ('BK002', 'The Gatsby', 150, 'Fitzgerald', 'fiction', 29.99, '2011-05-25', 'PUB002'),
80          ('BK003', 'Mockingbird', 200, 'Harper', 'fiction', 24.50, '2015-08-10', 'PUB003'),
81          ('BK004', '1984', 180, 'George', 'dystopian', 19.99, '2014-02-28', 'PUB004'),
82          ('BK005', 'Catcher in the Rye', 120, 'Salinger', 'fiction', 15.75, '2012-09-14', 'PUB005'),
83          ('BK006', 'Pride and Prejudice', 250, 'Jane', 'romance', 34.50, '2016-11-30', 'PUB006'),
84          ('BK007', 'The Hobbit', 180, 'Tolkien', 'fantasy', 42.25, '2013-07-22', 'PUB007'),
85          ('BK008', 'Da Vinci Code', 200, 'Brown', 'mystery', 27.90, '2017-03-10', 'PUB008'),
86          ('BK009', 'Lord of the Rings', 300, 'Tolkien', 'fantasy', 55.00, '2018-06-05', 'PUB009'),
87          ('BK010', 'The Alchemist', 150, 'Coelho', 'fiction', 18.95, '2019-09-20', 'PUB010');
```

The output

Book_ID	Title	Quantity_In_Stock	Author	Genre	Price	Production_Date	Pub_ID
BK001	HP Deathly Hallows	235	Rowling	fantasy	59.9	2010-11-19	PUB001
BK002	The Gatsby	150	Fitzgerald	fiction	29.99	2011-05-25	PUB002
BK003	Mockingbird	200	Harper	fiction	24.5	2015-08-10	PUB003
BK004	1984	180	George	dystopian	19.99	2014-02-28	PUB004
BK005	Catcher in the Rye	120	Salinger	fiction	15.75	2012-09-14	PUB005
BK006	Pride and Prejudice	250	Jane	romance	34.5	2016-11-30	PUB006
BK007	The Hobbit	180	Tolkien	fantasy	42.25	2013-07-22	PUB007
BK008	Da Vinci Code	200	Brown	mystery	27.9	2017-03-10	PUB008
BK009	Lord of the Rings	300	Tolkien	fantasy	55	2018-06-05	PUB009
BK010	The Alchemist	150	Coelho	fiction	18.95	2019-09-20	PUB010
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

CREATE TABLE Feedback (...);:

insert into Feedback values (...);:

```
91      -- creates a table named Feedback
92      -- table has columns for Member_ID, Book_ID, Title, Rating, and Comment.
93      -- The Member_ID and Book_ID are designated as the foreign key.
94      CREATE TABLE Feedback (
95          Member_ID char(10),
96          FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
97          Book_ID char(5),
98          FOREIGN KEY (Book_ID) REFERENCES Book(Book_ID),
99          Title varchar(50),
100         Rating INT,
101         Comment VARCHAR(100)
102         );
103
104     -- command inserts data into the 'Feedback' table.
105     insert into Feedback values
106         ('MBR001', 'BK006', 'Pride and Prejudice', 8, 'Interesting plot, but the ending left me wanting more'),
107         ('MBR002', 'BK006', 'Pride and Prejudice', 10, 'Absolutely loved it! The characters are well-developed.'),
108         ('MBR003', 'BK010', 'The Alchemist', 7, 'Good read, but some parts were a bit slow'),
109         ('MBR004', 'BK008', 'Da Vinci Code', 8, 'A page-turner! Couldn\'t put it down until I finished.'),
110         ('MBR005', 'BK010', 'The Alchemist', 9, 'Captivating storyline, highly recommend!'),
111         ('MBR006', 'BK007', 'The Hobbit', 10, 'Absolutely loved it! The characters are well-developed.'),
112         ('MBR007', 'BK010', 'The Alchemist', 9, 'Well-written and thought-provoking'),
113         ('MBR008', 'BK005', 'Catcher in the Rye', 7, 'Not my cup of tea, but some might find it intriguing'),
114         ('MBR009', 'BK002', 'The Gatsby', 5, 'Expected more from the hype, a bit disappointed'),
115         ('MBR010', 'BK002', 'The Gatsby', 6, 'Mixed feelings about the characters, but the plot was decent');
```

The output

Member_ID	Book_ID	Title	Rating	Comment
MBR001	BK006	Pride and Prejudice	8	Interesting plot, but the ending left me wanting...
MBR002	BK006	Pride and Prejudice	10	Absolutely loved it! The characters are well-dev...
MBR003	BK010	The Alchemist	7	Good read, but some parts were a bit slow
MBR004	BK008	Da Vinci Code	8	A page-turner! Couldn't put it down until I finish...
MBR005	BK010	The Alchemist	9	Captivating storyline, highly recommend!
MBR006	BK007	The Hobbit	10	Absolutely loved it! The characters are well-dev...
MBR007	BK010	The Alchemist	9	Well-written and thought-provoking
MBR008	BK005	Catcher in the Rye	7	Not my cup of tea, but some might find it intrigu...
MBR009	BK002	The Gatsby	5	Expected more from the hype, a bit disappointed
MBR010	BK002	The Gatsby	6	Mixed feelings about the characters, but the pl...

CREATE TABLE Manager (...);:

INSERT INTO Manager VALUES (...);:

```
119     -- creates a table named Feedback
120     -- table has columns for ManagerID, First_Name, Email, and Address.
121     -- The ManagerID is designated as the primary key.
122     CREATE TABLE Manager (
123         ManagerID char(5) primary key,
124         First_Name varchar(50),
125         Email varchar(50),
126         Address varchar(50)
127         );
128
129     -- command inserts data into the 'Manager' table.
130     INSERT INTO Manager VALUES
131         ('MGR01','Safawi','safawi02@gmail.com','Semenyih'),
132         ('MGR02','Lim','Lim@gmail.com','Bangi'),
133         ('MGR03','Sathis','satis1@gmail.com','Serdang'),
134         ('MGR04','Wong','wongg@gmail.com','Ampang'),
135         ('MGR05','Nizam','nizam@gmail.com','Shah Alam'),
136         ('MGR06','Cheng','chengcute@gmail.com','Petaling Jaya'),
137         ('MGR07','Previn','previn08@gmail.com','Damansara'),
138         ('MGR08','Amri','amrii@gmail.com','Bukit Jalil'),
139         ('MGR09','Ain','ain@gmail.com','Kajang'),
140         ('MGR10','Eilee','eilee@gmail.com','Cheras');
```

The output

ManagerID	First_Name	Email	Address
MGR01	Safawi	safawi02@gmail.com	Semenyih
MGR02	Lim	Lim@gmail.com	Bangi
MGR03	Sathis	satis1@gmail.com	Serdang
MGR04	Wong	wongg@gmail.com	Ampang
MGR05	Nizam	nizam@gmail.com	Shah Alam
MGR06	Cheng	chengcute@gmail.com	Petaling Jaya
MGR07	Previn	previn08@gmail.com	Damansara
MGR08	Amri	amrii@gmail.com	Bukit Jalil
MGR09	Ain	ain@gmail.com	Kajang
MGR10	Eilee	eilee@gmail.com	Cheras
NULL	NULL	NULL	NULL

CREATE TABLE BookOrder (...);:

INSERT INTO BookOrder VALUES (...);:

```
144     -- creates a table named BookOrder
145     -- table has columns for Order_ID, Manager_ID, Member_ID, Book_ID, Order_Date, and Quantity.
146     -- The Order_ID is designated as the primary key.
147     CREATE TABLE BookOrder (
148         Order_ID char(5) PRIMARY KEY,
149         Manager_ID char(5),
150         FOREIGN KEY (Manager_ID) REFERENCES Manager(ManagerID),
151         Member_ID char(10),
152         FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
153         Book_ID char(5),
154         FOREIGN KEY (Book_ID) REFERENCES Book(Book_ID),
155         Order_Date DATE,
156         Quantity INT
157     );
158
159     -- command inserts data into the 'BookOrder' table.
160     INSERT INTO BookOrder VALUES
161     ('BO001', NULL, 'MBR001','BK001', '2023-01-05', 5),
162     ('BO002', 'MGR02', NULL,'BK002', '2023-01-10', 3),
163     ('BO003', 'MGR03', NULL,'BK003', '2023-02-15', 2),
164     ('BO004', 'MGR04', NULL, 'BK004', '2023-03-20', 7),
165     ('BO005', NULL, 'MBR004', 'BK005', '2023-04-25', 4),
166     ('BO006', NULL, 'MBR005','BK006', '2023-05-30', 6),
167     ('BO007', 'MGR07', NULL,'BK007', '2023-06-05', 8),
168     ('BO008', NULL, 'MBR007','BK008', '2023-07-10', 1),
169     ('BO009', 'MGR09', NULL,'BK009', '2023-08-15', 10),
170     ('BO010', 'MGR10', 'MBR010','BK010', '2023-09-20', 3);
```

The output

Order_ID	Manager_ID	Member_ID	Book_ID	Order_Date	Quantity
BO001	NULL	MBR001	BK001	2023-01-05	5
BO002	MGR02	NULL	BK002	2023-01-10	3
BO003	MGR03	NULL	BK003	2023-02-15	2
BO004	MGR04	NULL	BK004	2023-03-20	7
BO005	NULL	MBR004	BK005	2023-04-25	4
BO006	NULL	MBR005	BK006	2023-05-30	6
BO007	MGR07	NULL	BK007	2023-06-05	8
BO008	NULL	MBR007	BK008	2023-07-10	1
BO009	MGR09	NULL	BK009	2023-08-15	10
BO010	MGR10	MBR010	BK010	2023-09-20	3
NULL	NULL	NULL	NULL	NULL	NULL

CREATE TABLE DeliveryStatus (...);:

INSERT INTO DeliveryStatus VALUES (...);:

```
174     -- creates a table named DeliveryStatus
175     -- table has columns for Order_ID, Delivery_Date, and Status.
176     -- The Order_ID is designated as the foreign key.
177     CREATE TABLE DeliveryStatus (
178         Order_ID char(5),
179         FOREIGN KEY (Order_ID) REFERENCES BookOrder(Order_ID),
180         Delivery_Date DATE,
181         Status varchar(20)
182     );
183
184     -- command inserts data into the 'DeliveryStatus' table.
185     INSERT INTO DeliveryStatus VALUES
186     ('BO001', '2023-10-25', 'Delivered'),
187     ('BO002', '2023-01-08', 'In Transit'),
188     ('BO003', '2023-02-12', 'Delivered'),
189     ('BO004', '2023-03-18', 'Pending'),
190     ('BO005', '2023-04-22', 'Delivered'),
191     ('BO006', '2023-05-28', 'Delivered'),
192     ('BO007', '2023-07-03', 'In Transit'),
193     ('BO008', '2023-08-12', 'Pending'),
194     ('BO009', '2023-09-18', 'Delivered'),
195     ('BO010', '2023-10-23', 'In Transit');
```

The output

Order_ID	Delivery_Date	Status
BO001	2023-10-25	Delivered
BO002	2023-01-08	In Transit
BO003	2023-02-12	Delivered
BO004	2023-03-18	Pending
BO005	2023-04-22	Delivered
BO006	2023-05-28	Delivered
BO007	2023-07-03	In Transit
BO008	2023-08-12	Pending
BO009	2023-09-18	Delivered
BO010	2023-10-23	In Transit

information_schema.columns table.

```
199     -- using the information_schema.columns table,
200     -- which holds metadata about columns in all tables.
201     -- filters the results for tables in the 'apuebookstore' database,
202     -- (WHERE table_schema = 'apuebookstore').
203     -- selected columns are table_name (representing the name of the table) and,
204     -- column_name (representing the name of each column within the table).
205     SHOW TABLES FROM apuebookstore;
206     SELECT table_name, column_name
207     FROM information_schema.columns
208     WHERE table_schema = 'apuebookstore';
```

The output

TABLE_NAME	COLUMN_NAME
publisher	Address
publisher	Email
publisher	First_Name
publisher	Last_Name
publisher	Pub_ID
member	Address
member	Contact_Number
member	Customer_ID
member	Email
member	First_Name
member	Last_Name
member	Member_ID
manager	Address
manager	Email
manager	First_Name
manager	ManagerID
feedback	Book_ID
feedback	Comment
feedback	Member_ID
feedback	Rating
feedback	Title
deliverystatus	Delivery_Date
deliverystatus	Order_ID
deliverystatus	Status
bookorder	Book_ID
bookorder	Manager_ID
bookorder	Member_ID
bookorder	Order_Date
bookorder	Order_ID
bookorder	Quantity
book	Author
book	Book_ID
book	Genre
book	Price
book	Production_Date
book	Pub_ID
book	Quantity_In_Stock
book	Title

7. SQL – Data Manipulation Language

i. Find the total number of feedback per book. Show book id, book title, and total number of feedback per book.

```
209     -- i. 'Total Number of Feedback'
210 •  select Book_ID, Title, count(Rating) 'Total Number of Feedback' -- select Book_ID, Title, and count Rating
211     from Feedback -- from Feedback table
212     group by Book_ID, Title; -- group by Book_ID, Book_Title to get feedback per book
```

The output

	Book_ID	Title	Total Number of Feedback
▶	BK006	Pride and Prejudice	2
	BK010	The Alchemist	3
	BK008	Da Vinci Code	1
	BK007	The Hobbit	1
	BK005	Catcher in the Rye	1
	BK002	The Gatsby	2

ii. Find the total number of feedback per member. Show member id, member name, and total number of feedback per member.

```
214     -- ii. total number of feedback per member.
215     -- Select Member_ID, First_Name, and count of Ratings as 'Total Number of Feedback' for each member
216 •  SELECT f.Member_ID, m.First_Name, COUNT(Rating) AS 'Total Number of Feedback'
217     -- From the Feedback table (aliased as f) and the Member table (aliased as m)
218     FROM Feedback AS f
219     -- Join the tables based on Member_ID
220     INNER JOIN Member AS m ON f.Member_ID = m.Member_ID
221     -- Group the results by Member_ID and First_Name to get feedback count per member
222     GROUP BY f.Member_ID, m.First_Name;
```

The output

	Member_ID	First_Name	Total Number of Feedback
▶	MBR001	Khan	1
	MBR002	Aisha	1
	MBR003	John	1
	MBR004	Siti	1
	MBR005	David	1
	MBR006	Kogee	1
	MBR007	Ahmad	1
	MBR008	Nina	1
	MBR009	Tina	1
	MBR010	Linda	1

iii. Find the total number of book published by each publisher. Show publisher id, publisher name, and number of book published.

```
224     -- iii. Find the total number of book published by each publisher
225     -- Select Pub_ID, First_Name, and count of Book_ID as 'Total Number of Books Published' for each publisher
226 •   SELECT p.Pub_ID, p.First_Name, COUNT(b.Book_ID) AS 'Total Number of Books Published'
227     -- From the Book table (aliased as b) and the Publisher table (aliased as p)
228     FROM Book AS b
229     -- Join the tables based on Pub_ID
230     INNER JOIN Publisher AS p
231     ON b.Pub_ID = p.Pub_ID
232     -- Group the results by Pub_ID and First_Name to get the count of books published per publisher
233     GROUP BY p.Pub_ID, p.First_Name;
```

The output

	Pub_ID	First_Name	Total Number of Books Published
▶	PUB001	Ong	1
	PUB002	Tan	1
	PUB003	Wan	1
	PUB004	Ng	1
	PUB005	Karisma	1
	PUB006	Goh	1
	PUB007	Junaida	1
	PUB008	Lee	1
	PUB009	Amirah	1
	PUB010	Peri	1

iv. Find the total number of book for each genre.

```
235     -- iv. Find the total number of book for each genre.
236     -- Select Genre and count of Book_ID as 'Total Number of Books"
237 •   select Genre, count(Book_ID) as 'Total Number of Book'
238     from Book -- from Book table
239     group by Genre; -- group by Book_Category to get book for each category
```

The output

	Genre	Total Number of Book
▶	fantasy	3
	fiction	4
	dystopian	1
	romance	1
	mystery	1

v. From the book table, list the books where quantity is more than the average quantity of all books.

```
240     -- v. list the books where quantity is more than the average quantity of all books.
241     -- Select all columns from the Book table
242 •   SELECT *
243     FROM Book
244     -- Where the Quantity_In_Stock is greater than the average quantity of all books
245     WHERE Quantity_In_Stock > (SELECT AVG(Quantity_In_Stock) FROM Book);
```

The output

Book_ID	Title	Quantity_In_Stock	Author	Genre	Price	Production_Date	Pub_ID
BK001	HP Deathly Hallows	235	Rowling	fantasy	59.9	2010-11-19	PUB001
BK003	Mockingbird	200	Harper	fiction	24.5	2015-08-10	PUB003
BK006	Pride and Prejudice	250	Jane	romance	34.5	2016-11-30	PUB006
BK008	Da Vinci Code	200	Brown	mystery	27.9	2017-03-10	PUB008
BK009	Lord of the Rings	300	Tolkien	fantasy	55	2018-06-05	PUB009
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

vi. Find the total number of books ordered by store manager from various publishers.

```
248     -- vi. Find the total number of books ordered by store manager from various publishers.
249     -- Select specific columns to display in the result
250 •   SELECT
251         m.ManagerID,                -- Manager ID
252         m.First_Name,               -- Manager's first name
253         COUNT(bo.Book_ID) AS 'Total Books Ordered' -- Count of unique books ordered by the manager
254     FROM
255         Manager m                   -- From the Manager table (aliased as m)
256     JOIN
257         BookOrder bo ON m.ManagerID = bo.Manager_ID -- Join with BookOrder based on Manager ID
258     GROUP BY
259         m.ManagerID, m.First_Name;  -- Group the results by Manager ID and first name
```

The output

ManagerID	First_Name	Total Books Ordered
MGR02	Lim	1
MGR03	Sathis	1
MGR04	Wong	1
MGR07	Previn	1
MGR09	Ain	1
MGR10	Eilee	1

vii. Show the members who did not make any order

```
262     -- vii. Show the members who did not make any order
263     -- Select Member_ID, First_Name, and Last_Name from the Member table
264 •   SELECT M.Member_ID, M.First_Name, M.Last_Name
265     FROM Member M
266     -- Left join with BookOrder based on Member_ID
267     LEFT JOIN BookOrder BO ON M.Member_ID = BO.Member_ID
268     -- Filter the results to include only rows where there is no corresponding order (BO.Quantity IS NULL)
269     WHERE BO.Quantity IS NULL;
```

The output

	Member_ID	First_Name	Last_Name
▶	MBR002	Aisha	Lee
	MBR003	John	Doe
	MBR006	Kogee	Vathi
	MBR008	Nina	Lim
	MBR009	Tina	Rajah

viii. Find the genres of the book which has the most number of quantity in stock.

```
271     -- viii. Find the genres of the book which has the most number of quantity in stock
272     -- Select the genre and the total quantity in stock for each genre
273 •   SELECT Genre, SUM(Quantity_In_Stock) AS 'Total Quantity'
274     -- From the Book table
275     FROM Book
276     -- Group the results by genre
277     GROUP BY Genre
278     -- Filter the results to include only genres with a total quantity equal to the maximum total quantity
279     -- Subquery: Find the maximum total quantity among all genres
280     -- Calculate the total quantity for each genre
281     HAVING SUM(Quantity_In_Stock) = (SELECT MAX(TotalQuantity) FROM (SELECT SUM(Quantity_In_Stock)
282     AS TotalQuantity FROM Book GROUP BY Genre)
283     AS GenreQuantities);
```

The output

	Genre	Total Quantity
▶	fantasy	715

ix. A list of purchased books that have not been delivered to members. The list should show member identification number, address, contact number, book serial number, book title, quantity, date and status of delivery.

```
284     -- ix. A list of purchased books that have not been delivered
285     -- Select specific columns to display in the result
286 •   SELECT
287         M.Member_ID,
288         M.Address,
289         M.Contact_Number,
290         BO.Book_ID,
291         B.Title AS Book_Title,
292         BO.Quantity,
293         BO.Order_Date,
294         DS.Status
295     FROM
296         Member M  -- From the Member table (aliased as M)
297         JOIN BookOrder BO ON M.Member_ID = BO.Member_ID -- Join with BookOrder based on Member ID
298         JOIN Book B ON BO.Book_ID = B.Book_ID -- Join with Book based on Book ID
299         LEFT JOIN DeliveryStatus DS ON BO.Order_ID = DS.Order_ID -- Left join with DeliveryStatus based on Order ID
300     -- Filter the results to include only rows where the delivery status is NULL or 'Pending'
301     WHERE
302         DS.Status IS NULL OR DS.Status = 'Pending';
```

The output

	Member_ID	Address	Contact_Number	Book_ID	Book_Title	Quantity	Order_Date	Status
▶	MBR007	Klang	016-5432109	BK008	Da Vinci Code	1	2023-07-10	Pending

x. Show the members who made more than 2 orders.

```
306     -- x. Show the members who made more than 2 orders
307     -- Select specific columns from the BookOrder and Member tables
308 •   SELECT bo.Order_ID, bo.Member_ID, m.First_Name, bo.Quantity
309     -- From the BookOrder table (aliased as bo) and the Member table (aliased as m)
310     FROM BookOrder bo
311     JOIN Member m ON bo.Member_ID = m.Member_ID
312     -- Filter the results to include only rows where the order quantity is greater than 2
313     WHERE bo.Quantity > 2;
```

The output

	Order_ID	Member_ID	First_Name	Quantity
▶	BO001	MBR001	Khan	5
	BO005	MBR004	Siti	4
	BO006	MBR005	David	6
	BO010	MBR010	Linda	3

8. References

1. Advantages of Database Management System. (2019, November 20). *GeeksforGeeks*. https://www.geeksforgeeks.org/advantages-of-database-management-system/
2. *Create a diagram with crow's foot database notation—Microsoft Support*. (n.d.). Retrieved December 14, 2023, from https://support.microsoft.com/en-au/office/create-a-diagram-with-crow-s-foot-database-notation-1ec22af9-3bd3-4354-b2b5-ed5752af6769
3. *Defining and Establishing Business Rules: Chapter 11. Business Rules: Part II: The Design Process: Database design for mere mortals: SQL :: eTutorials.org*. (n.d.). Retrieved December 15, 2023, from https://etutorials.org/SQL/Database+design+for+mere+mortals/Part+II+The+Design+Process/Chapter+11.+Business+Rules/Defining+and+Establishing+Business+Rules/
4. Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. (2012). A Survey and Comparison of Relational and Non-Relational Database. *International Journal of Engineering Research & Technology*, *1*(6). https://doi.org/10.17577/IJERTV1IS6024
5. *MySQL :: MySQL Tutorial: 7 Examples of Common Queries*. (n.d.). Retrieved December 15, 2023, from https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.2/en/examples.html