CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
Autonomous Institute | Affiliated to Osmania University

Online Bookstore

Presented By:
Amballa Sneha (1601-23-733-001)
Amireddy Shriya Reddy (1601-23-733-002)
Amukta Malya Yajamanyam (1601-23-733-003)

Lab: Database Systems Lab
Course Code: 22CSC13
Submitted To: Dr. Anila M,
CSE Department

Problem Statement

- Online Bookstores are facing **operational inefficiencies** that hinder customer satisfaction and overall performance. A streamlined database solution is essential to address these challenges.

- **Problems Faced:**

- 1. **Inefficient Management**: Challenges in organizing and managing data related to books, authors, and publishers.

- 2. **Limited Search Functionality**: Insufficient search and browsing features make it difficult for customers to find specific titles or genres.

- 3. **Cumbersome Order Processing**: Complex order placement processes result in abandoned shopping carts and lost sales.

- 4. **Poor Inventory Tracking**: Inaccurate stock level tracking leads to overselling or stockouts, negatively affecting customer satisfaction.

Objectives & Motivation

- Manage essential information about books, authors, publishers, customers, orders, and inventory.
- Enable easy searching, browsing, and ordering for a better user experience.
- Ensure accurate stock levels and availability information for customers.
- Streamline order processing and tracking to enhance customer satisfaction.
- Provide reporting tools for insights into sales trends and customer preferences.
- Maintain high data quality and integrity through constraints and validation rules.

Entity-Relationship Diagram

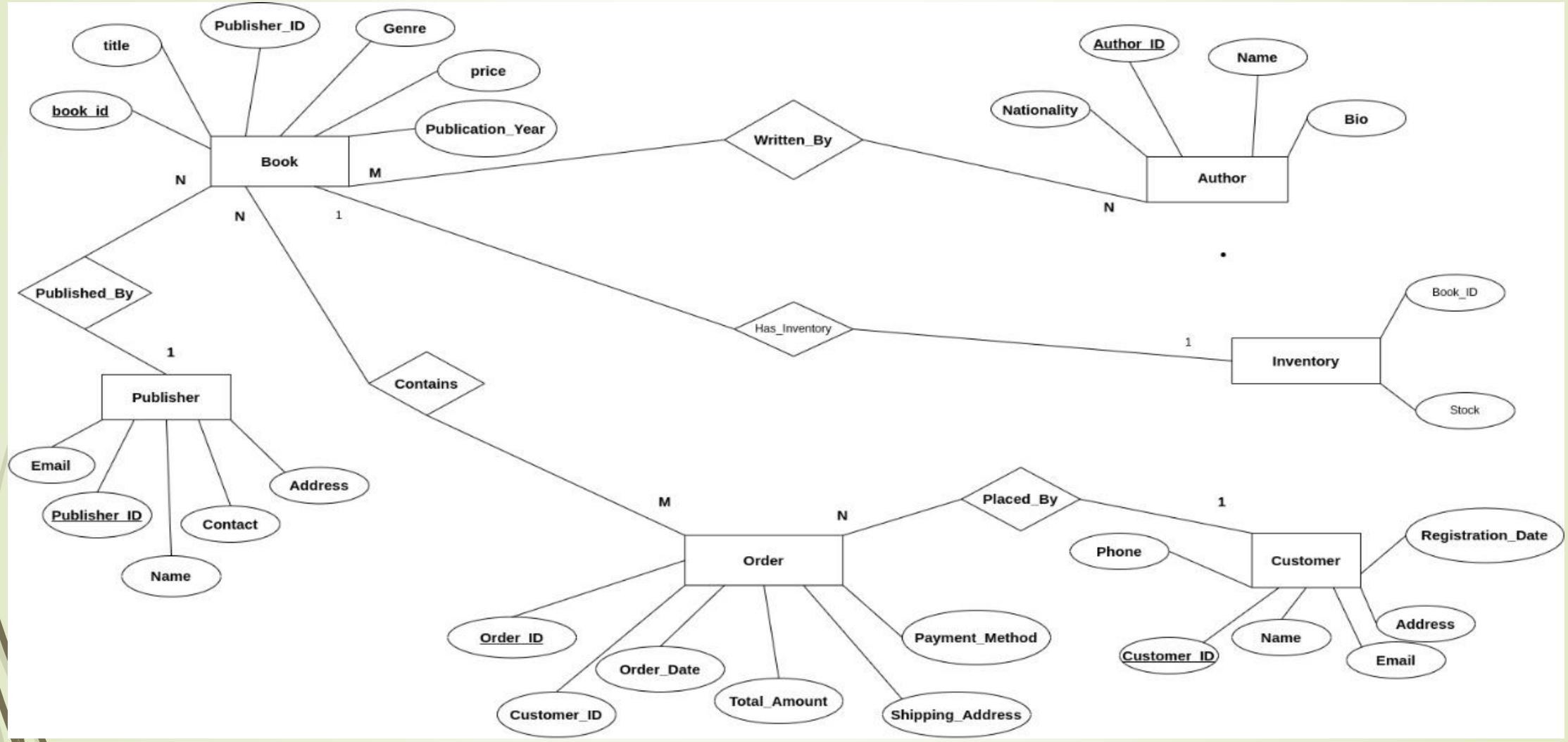

Fig-1: Online Bookstore ER Diagram

Entities and their Attributes

1.Book Entity

The Book entity represents the books available in the online bookstore with the following attributes:

- **Book_ID:** Primary key (uniquely identifies each book)
- **Title:** The name of the book
- **Author_ID:** Foreign key referencing the Author entity
- **Publisher_ID:** Foreign key referencing the Publisher entity
- **Publication_Year:** The year the book was published
- **Price:** The selling price of the book
- **Genre:** The category or genre of the book

2. Author Entity

The Author entity represents the authors of the books with the following attributes:

- **Author_ID:** Primary key (uniquely identifies each author)
- **Name:** The full name of the author
- **Nationality:** The nationality of the author
- **Bio:** A brief biography of the author

3. Publisher Entity

The Publisher entity represents the publishing companies of the books with the following attributes:

- **Publisher_ID:** Primary key (uniquely identifies each publisher)
- **Name:** The name of the publishing company
- **Address:** The address of the publisher
- **Contact:** Contact information for the publisher
- **Email:** Email address of the publisher

4. Customer Entity

The Customer entity represents the individuals who have accounts and place orders with the online bookstore, with the following attributes:

- **Customer_ID:** Primary key (uniquely identifies each customer)
- **Name:** The full name of the customer
- **Address:** The mailing address of the customer
- **Email:** The email address of the customer (unique)
- **Phone:** The phone number of the customer
- **Registration_Date:** The date when the customer registered

Entities and their Attributes

5. Order Entity

The Order entity represents the purchase orders placed by customers with the following attributes:

- **Order_ID:** Primary key (uniquely identifies each order)
- **Customer_ID:** Foreign key referencing the Customer entity
- **Order_Date:** The date when the order was placed
- **Total_Amount:** The total cost of the order
- **Payment_Method:** The method of payment used for the order
- **Shipping_Address:** The address where the order will be shipped

6. Inventory Entity

The Inventory entity represents the stock levels of each book in the bookstore with the following attributes:

- **Book_ID:** Primary key and foreign key referencing the Book entity (uniquely identifies the book in the inventory)
- **Stock:** The current number of copies of the book in stock

Relationship between Entities

Book Entity
Description: The Book table is a strong entity that stores all information about the books available for order.
Attributes:
• book id: Primary Key that uniquely identifies each book.
• title: Title of the book.
• Publisher ID: Foreign Key linking to the Publisher table.
• Genre: Category or genre of the book.
• price: Price of the book.
• Publication Year: Year in which the book was published.
Relationships:
• Related to the Author entity through the Written By relationship (Many-to-Many).
• Connected to the Publisher entity through the Published By relationship (Many-to-One).
• Linked to the Inventory entity through the Has Inventory relationship (One-to-One).
• Related to the Order entity through the Contains relationship (Many-to-Many).
Author Entity
Description: The Author table is a strong entity that stores information about authors who write books.
Attributes:
• Author ID: Primary Key that uniquely identifies each author.
• Name: Name of the author.
• Nationality: Nationality of the author.
• Bio: Short biography of the author.
Relationships:
• Connected to the Book entity through the Written By relationship (Many-to-Many).
Publisher Entity
Description: The Publisher table is a strong entity that manages details about book publishers.
Attributes:
• Publisher ID: Primary Key for unique identification.
• Name: Name of the publisher.
• Email: Email address of the publisher.
• Contact: Contact number of the publisher.
• Address: Address of the publisher.
Relationships:
• Related to the Book entity through the Published By relationship (One-to-Many).

Relationship between Entities

Customer Entity
Description: The Customer table is a strong entity that stores information about customers placing orders.
Attributes:
• Customer ID: Primary Key that uniquely identifies each customer.
• Name: Name of the customer.
• Email: Email address of the customer.
• Phone: Contact number of the customer.
• Address: Address of the customer.
• Registration Date: Date the customer registered on the system.
Relationships:
• Related to the Order entity through the Placed By relationship (One-to-Many).
Order Entity
Description: The Order table is a strong entity that manages customer orders for books.
Attributes:
• Order ID: Primary Key that uniquely identifies each order.
• Order Date: Date when the order was placed.
• Total Amount: Total amount for the order.
• Customer ID: Foreign Key linking the order to the customer.
• Payment Method: Method used for payment (e.g., Credit Card, PayPal).
• Shipping Address: Address where the order should be delivered.
Relationships:
• Connected to the Customer entity through the Placed By relationship (Many-to-One).
• Linked to the Book entity through the Contains relationship (Many-to-Many).
Inventory Entity
Description: The Inventory table is a strong entity that keeps track of the stock of each book.
Attributes:
• Book ID: Primary Key (and Foreign Key linking to Book table).
• Stock: Quantity of the book available in the inventory.
Relationships:
• Connected to the Book entity through the Has Inventory relationship (One-to-One).

Database Schema

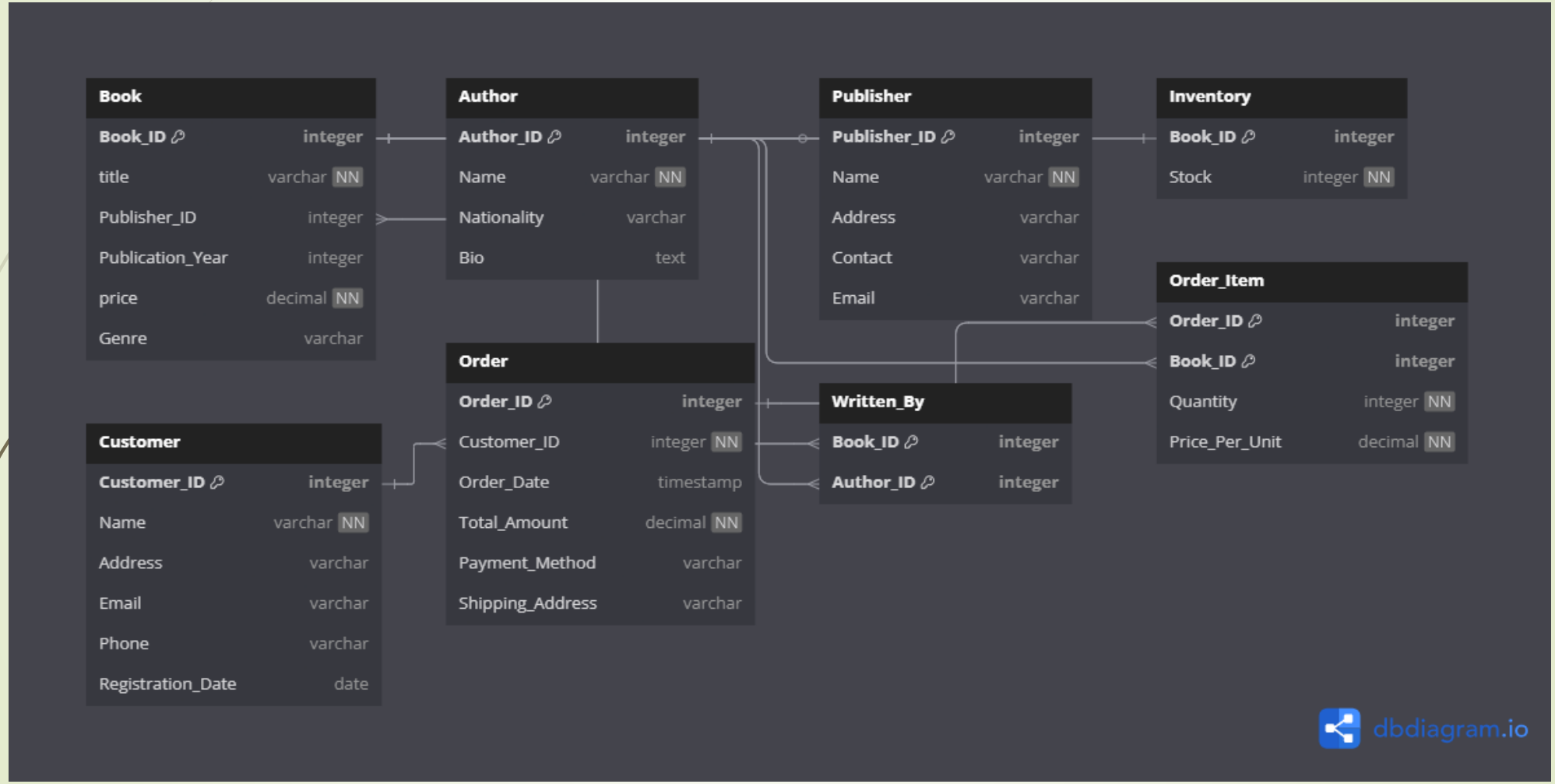

Fig-2: Diagram of Database Schema

Creation of Tables

Field	Type	Null	Key	Default	Extra
publisher_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	
contact	varchar(15)	YES		NULL	
address	varchar(255)	YES		NULL	

Field	Type	Null	Key	Default	Extra
author_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
nationality	varchar(50)	YES		NULL	
bio	text	YES		NULL	

1. **<u>Publisher Table:</u>**

```
CREATE TABLE Publisher (
    publisher_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    contact VARCHAR(15),
    address VARCHAR(255)
);
DESC Publisher;
```

2. **<u>Author Table</u>**:

```
CREATE TABLE Author (
    author_id INT PRIMARY KEY,
    name VARCHAR(100),
    nationality VARCHAR(50),
    bio TEXT
);

DESC Author;
```

Continuation

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	
title	varchar(150)	YES		NULL	
genre	varchar(50)	YES		NULL	
price	decimal(8,2)	YES		NULL	
publication_year	year	YES		NULL	
publisher_id	int	YES	MUL	NULL	

3. Book Table:

```
CREATE TABLE Book (
    book_id INT PRIMARY KEY,
    title VARCHAR(150),
    genre VARCHAR(50),
    price DECIMAL(8,2),
    publication_year YEAR,
    publisher_id INT,
    FOREIGN KEY (publisher_id) REFERENCES Publisher(publisher_id)
);

DESC Book;
```

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	
phone	varchar(15)	YES		NULL	
address	text	YES		NULL	
registration_date	date	YES		NULL	

4. Customer Table

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    address TEXT,
    registration_date DATE DEFAULT NULL
);

DESC Customer;
```

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	
stock	int	YES		NULL	

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
order_date	date	YES		NULL	
total_amount	decimal(10,2)	YES		NULL	
payment_method	varchar(50)	YES		NULL	
shipping_address	text	YES		NULL	

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	
author_id	int	NO	PRI	NULL	

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
book_id	int	NO	PRI	NULL	
quantity	int	YES		NULL	

5. Inventory Table:

```
CREATE TABLE Inventory (
   book_id INT PRIMARY KEY,
   stock INT CHECK (stock >= 0),
   FOREIGN KEY (book_id) REFERENCES Book(book_id)
);

DESC Inventory;
```

6. Order Table:

```
CREATE TABLE `Order` (
   order_id INT PRIMARY KEY,
   customer_id INT,
   order_date DATE,
   total_amount DECIMAL(10,2),
   payment_method VARCHAR(50),
   shipping_address TEXT,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

DESC `Order`;
```

7. Written By Table:

```
CREATE TABLE Written_By (
   book_id INT,
   author_id INT,
   PRIMARY KEY (book_id, author_id),
   FOREIGN KEY (book_id) REFERENCES Book(book_id),
   FOREIGN KEY (author_id) REFERENCES Author(author_id)
);
DESC Written_By;
```

8. Contains Table:

```
CREATE TABLE Contains (
   order_id INT,
   book_id INT,
   quantity INT,
   PRIMARY KEY (order_id, book_id),
   FOREIGN KEY (order_id) REFERENCES `Order`(order_id),
   FOREIGN KEY (book_id) REFERENCES Book(book_id)
);

DESC Contains;
```

Data Insertion

	PUBLISHER_ID	NAME	EMAIL	CONTACT	ADDRESS
1	1	Pearson	contact@pearson.co	1234567890	New York
2	2	Penguin	support@penguin.cc	9876543210	London
3	3	HarperCollins	info@harper.com	8888888888	Toronto
4	4	Macmillan	help@macmillan.con	7777777777	Berlin
5	5	Scholastic	scholastic@edu.com	6666666666	Delhi

	AUTHOR_ID	NAME	NATIONALITY	BIO
1	101	J.K. Rowling	British	Author of Harry Potte
2	102	George Orwell	British	Author of 1984
3	103	Dan Brown	American	Author of Da Vinci Cc
4	104	Agatha Christie	British	Mystery novelist
5	105	Chetan Bhagat	Indian	Contemporary noveli

```
-- Publishers
INSERT INTO Publisher VALUES
(1, 'Pearson', 'contact@pearson.com', '1234567890', 'New York');
INSERT INTO Publisher VALUES
(2, 'Penguin', 'support@penguin.com', '9876543210', 'London');
INSERT INTO Publisher VALUES
(3, 'HarperCollins', 'info@harper.com', '8888888888', 'Toronto');
INSERT INTO Publisher VALUES
(4, 'Macmillan', 'help@macmillan.com', '7777777777', 'Berlin');
INSERT INTO Publisher VALUES
(5, 'Scholastic', 'scholastic@edu.com', '6666666666', 'Delhi');

-- Authors
INSERT INTO Author VALUES
(101, 'J.K. Rowling', 'British', 'Author of Harry Potter');
INSERT INTO Author VALUES
(102, 'George Orwell', 'British', 'Author of 1984');
INSERT INTO Author VALUES
(103, 'Dan Brown', 'American', 'Author of Da Vinci Code');
INSERT INTO Author VALUES
(104, 'Agatha Christie', 'British', 'Mystery novelist');
INSERT INTO Author VALUES
(105, 'Chetan Bhagat', 'Indian', 'Contemporary novelist');

SELECT * FROM Publisher;
SELECT * FROM Author;
```

	BOOK_ID	TITLE	GENRE	PRICE	PUBLICATION_YEAF	PUBLISHER_ID
1	6	New Book	Thriller	300	2024	1
2	1	Harry Potter	Fantasy	499.99	2001	1
3	2	1984	Dystopian	299.99	1949	2
4	3	Da Vinci Code	Thriller	499	2003	3
5	4	Murder on the Orient	Mystery	350.5	1934	4
6	5	2 States	Romance	250	2009	5

	BOOK_ID	STOCK
1	1	10
2	2	5
3	3	7
4	4	3
5	5	8

	CUSTOMER_ID	NAME	EMAIL	PHONE	ADDRESS	REGISTRATION_DAT
1	201	Sneha	sneha@example.con	9000000001	Hyderabad	1/1/2025, 12:00:00
2	202	Shriya	shriya@example.con	9000000002	Mumbai	1/5/2025, 12:00:00
3	203	Amukta	amukta@example.cc	9000000003	Delhi	1/10/2025, 12:00:0
4	204	Sravani	sravani@example.co	9000000004	New York	1/15/2025, 12:00:0
5	205	Mounika	mounika@example.c	9000000005	London	1/20/2025, 12:00:0

```
-- Books
INSERT INTO Book VALUES
(1, 'Harry Potter', 'Fantasy', 399.99, 2001, 1);
INSERT INTO Book VALUES
(2, '1984', 'Dystopian', 299.99, 1949, 2);
INSERT INTO Book VALUES
(3, 'Da Vinci Code', 'Thriller', 499.00, 2003, 3);
INSERT INTO Book VALUES
(4, 'Murder on the Orient Express', 'Mystery', 350.50, 1934, 4);
INSERT INTO Book VALUES
(5, '2 States', 'Romance', 250.00, 2009, 5);

-- Inventory
INSERT INTO Inventory VALUES (1, 10);
INSERT INTO Inventory VALUES (2, 5);
INSERT INTO Inventory VALUES (3, 7);
INSERT INTO Inventory VALUES (4, 3);
INSERT INTO Inventory VALUES (5, 8);

-- Customers
INSERT INTO Customer VALUES
(201, 'Sneha', 'sneha@example.com', '9000000001', 'Hyderabad',
STR_TO_DATE('2025-01-01', '%Y-%m-%d'));
INSERT INTO Customer VALUES
(202, 'Shriya', 'shriya@example.com', '9000000002', 'Mumbai',
STR_TO_DATE('2025-01-05', '%Y-%m-%d'));
INSERT INTO Customer VALUES
(203, 'Amukta', 'amukta@example.com', '9000000003', 'Delhi',
STR_TO_DATE('2025-01-10', '%Y-%m-%d'));
INSERT INTO Customer VALUES
(204, 'Sravani', 'sravani@example.com', '9000000004', 'New York',
STR_TO_DATE('2025-01-15', '%Y-%m-%d'));
INSERT INTO Customer VALUES
(205, 'Mounika', 'mounika@example.com', '9000000005', 'London',
STR_TO_DATE('2025-01-20', '%Y-%m-%d'));

SELECT * FROM Book;
SELECT * FROM Inventory;
SELECT * FROM Customers;
```

	ORDER_ID	CUSTOMER_ID	ORDER_DATE	TOTAL_AMOUNT	PAYMENT_METHOD	SHIPPING_ADDRESS
1	301	201	4/1/2025, 12:00:00	649.99	UPI	Hyderabad
2	302	202	4/2/2025, 12:00:00	299.99	Card	Mumbai
3	303	203	4/3/2025, 12:00:00	499	Cash	Delhi
4	304	204	4/4/2025, 12:00:00	750.5	UPI	New York
5	305	205	4/5/2025, 12:00:00	250	Card	London

	BOOK_ID	AUTHOR_ID
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	101

	ORDER_ID	BOOK_ID	QUANTITY
1	301	1	1
2	302	2	1
3	303	3	1
4	304	4	2
5	305	5	1

```
-- Orders
INSERT INTO `Order` VALUES
(301, 201, STR_TO_DATE('2025-04-01', '%Y-%m-%d'), 649.99, 'UPI',
'Hyderabad');
INSERT INTO `Order` VALUES
(302, 202, STR_TO_DATE('2025-04-02', '%Y-%m-%d'), 299.99, 'Card',
'Mumbai');
INSERT INTO `Order` VALUES
(303, 203, STR_TO_DATE('2025-04-03', '%Y-%m-%d'), 499.00, 'Cash',
'Delhi');
INSERT INTO `Order` VALUES
(304, 204, STR_TO_DATE('2025-04-04', '%Y-%m-%d'), 750.50, 'UPI', 'New
York');
INSERT INTO `Order` VALUES
(305, 205, STR_TO_DATE('2025-04-05', '%Y-%m-%d'), 250.00, 'Card',
'London');

-- Written_By
INSERT INTO Written_By VALUES (1, 101);
INSERT INTO Written_By VALUES (2, 102);
INSERT INTO Written_By VALUES (3, 103);
INSERT INTO Written_By VALUES (4, 104);
INSERT INTO Written_By VALUES (5, 105);

-- Contains
INSERT INTO Contains VALUES (301, 1, 1);
INSERT INTO Contains VALUES (302, 2, 1);
INSERT INTO Contains VALUES (303, 3, 1);
INSERT INTO Contains VALUES (304, 4, 2);
INSERT INTO Contains VALUES (305, 5, 1);

SELECT * FROM `Order`;
SELECT * FROM Written_By;
SELECT * FROM Contains;
```

DEMONSTRATION OF GROUP BY, ORDER BY AND HAVING, WITH AGGREGATE FUNCTIONS

	PUBLISHERNAME	NUMBEROFBOOKS
1	Pearson	1
2	Penguin	1
3	HarperCollins	1
4	Macmillan	1
5	Scholastic	1

	GENRE	AVERAGEPRICE
1	Fantasy	499.99
2	Thriller	499
3	Mystery	350.5
4	Dystopian	299.99
5	Romance	250

	AUTHORNAME	NUMBEROFBOOKS
1	J.K. Rowling	2

1. Question (GROUP BY):
Form a query to retrieve each publisher's name along with the total count of books associated with them.

```
SELECT
   p.name AS PublisherName,
   COUNT(b.book_id) AS NumberOfBooks
FROM
   Publisher p
JOIN
   Book b ON p.publisher_id = b.publisher_id
GROUP BY
   p.name;
```

2. Question (ORDER BY):
Construct a query to obtain a list of book genres and their respective average prices, ordered from the highest average price to the lowest.

```
SELECT
   b.genre AS Genre,
   AVG(b.price) AS AveragePrice
FROM
   Book b
GROUP BY
   b.genre
ORDER BY
   AveragePrice DESC;
```

3. Question (HAVING):
Write a query to identify and retrieve the names of authors who have authored more than one book within the database.

```
SELECT
   a.name AS AuthorName,
   COUNT(wb.book_id) AS NumberOfBooks
FROM
   Author a
JOIN
   Written_By wb ON a.author_id = wb.author_id
GROUP BY
   a.name
HAVING
   NumberOfBooks > 1;
```

DEMONSTRATION OF TRIGGERS

```
SQL> INSERT INTO "Order" VALUES (307, 201, TO_DATE('2025-04-06','YYYY-MM-DD'), 499.99, 'UPI', 'Hyderabad')

New order placed. Order ID: 307

1 row inserted.

Elapsed: 00:00:00.006
```

```
SQL> INSERT INTO Contains VALUES (307, 1, 2)

Inventory updated for Book ID: 1

1 row inserted.

Elapsed: 00:00:00.002
```

Definition: Triggers are database objects that automatically execute a predefined set of SQL statements in response to specific events (like INSERT, UPDATE, or DELETE) on a table.

Purpose: Enforce business rules, maintain data integrity, audit changes, or automate tasks related to data modification.

Code and Demonstration Query:

```
CREATE OR REPLACE TRIGGER trg_update_inventory
AFTER INSERT ON Contains
FOR EACH ROW
BEGIN
   UPDATE Inventory
   SET stock = stock - :NEW.quantity
   WHERE book_id = :NEW.book_id;
   DBMS_OUTPUT.PUT_LINE('Inventory updated for Book ID: ' | |
:NEW.book_id);
END;
/

SELECT book_id, stock
FROM Inventory
WHERE book_id = 1;

INSERT INTO "Order"
VALUES (307, 201, STR_TO_DATE('2025-04-06','%Y-%m-%d'), 499.99, 'UPI',
'Hyderabad');
INSERT INTO Contains
VALUES (307, 1, 2);

SELECT book_id, stock
FROM Inventory
WHERE book_id = 1;
```

CURSORS: IMPLICT AND EXPLICIT

Implicit Cursor:

Definition: Automatically created by the database when a single SQL statement (such as INSERT, UPDATE, DELETE, or SELECT INTO) is executed.

Purpose: Simplifies small operations where you do not need to control the cursor manually.

Explicit Cursor:

Definition: Manually declared and controlled by the programmer to handle queries that return multiple rows.

Purpose: Gives detailed control over fetching, looping, and processing multiple rows one at a time.

DEMONSTRATION OF CURSORS

```
1 Fantasy book(s) price updated.
```

```
SQL> DECLARE
         CURSOR c_customer_cursor IS
             SELECT customer_id, name
             FROM Customer...
Show more...


Customer ID: 202 Name: Shriya


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.016
```

Implicit Cursor Example

```
DECLARE
   total_books NUMBER(3);
BEGIN
   UPDATE Book
   SET price = price + 50
   WHERE genre = 'Fantasy';

   IF SQL%NOTFOUND THEN
      DBMS_OUTPUT.PUT_LINE('No Fantasy books found.');
   ELSIF SQL%FOUND THEN
      total_books := SQL%ROWCOUNT;
      DBMS_OUTPUT.PUT_LINE(total_books || ' Fantasy book(s) price
updated.');
   END IF;
END;
/
```

Explicit Cursor Example

```
DECLARE
   CURSOR c_customer_cursor IS
      SELECT customer_id, name
      FROM Customer
      WHERE DBMS_LOB.SUBSTR(address, 100) = 'Mumbai';
   v_cust_id Customer.customer_id%TYPE;
   v_cust_name Customer.name%TYPE;
BEGIN
   OPEN c_customer_cursor;
   LOOP
      FETCH c_customer_cursor INTO v_cust_id, v_cust_name;
      EXIT WHEN c_customer_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_cust_id || ' Name: ' ||
v_cust_name);
   END LOOP;
   CLOSE c_customer_cursor;
END;
/
```

Conclusion

- The Online Bookstore database project successfully addressed the management of books, authors, publishers, customers, orders, and inventory. The ER diagram effectively modeled entities and relationships, and the database was implemented with DDL commands and constraints to ensure data integrity. SQL queries facilitated searching, browsing, and order processing, while triggers and cursors automated inventory updates and data handling. This robust system provides a strong foundation for online bookstore operations.

Thank you!