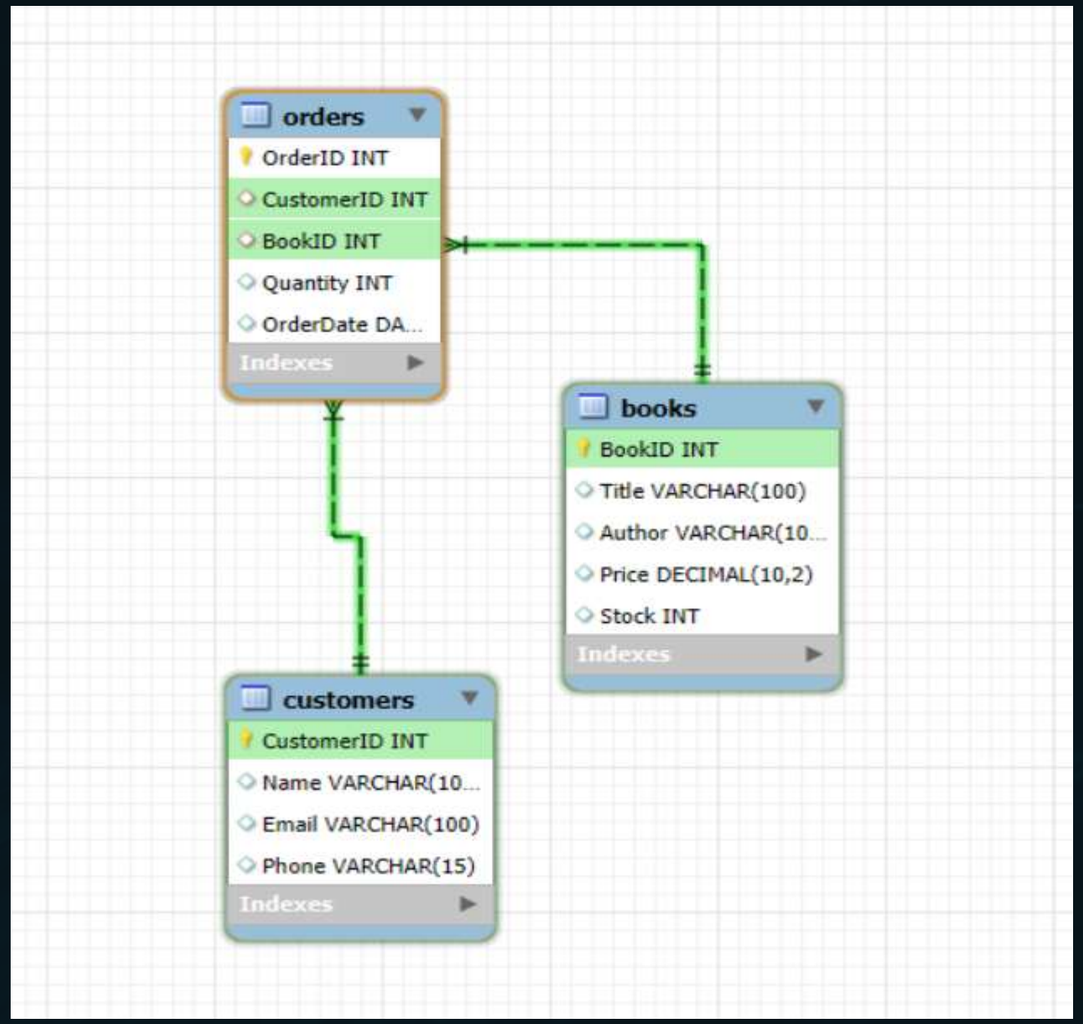
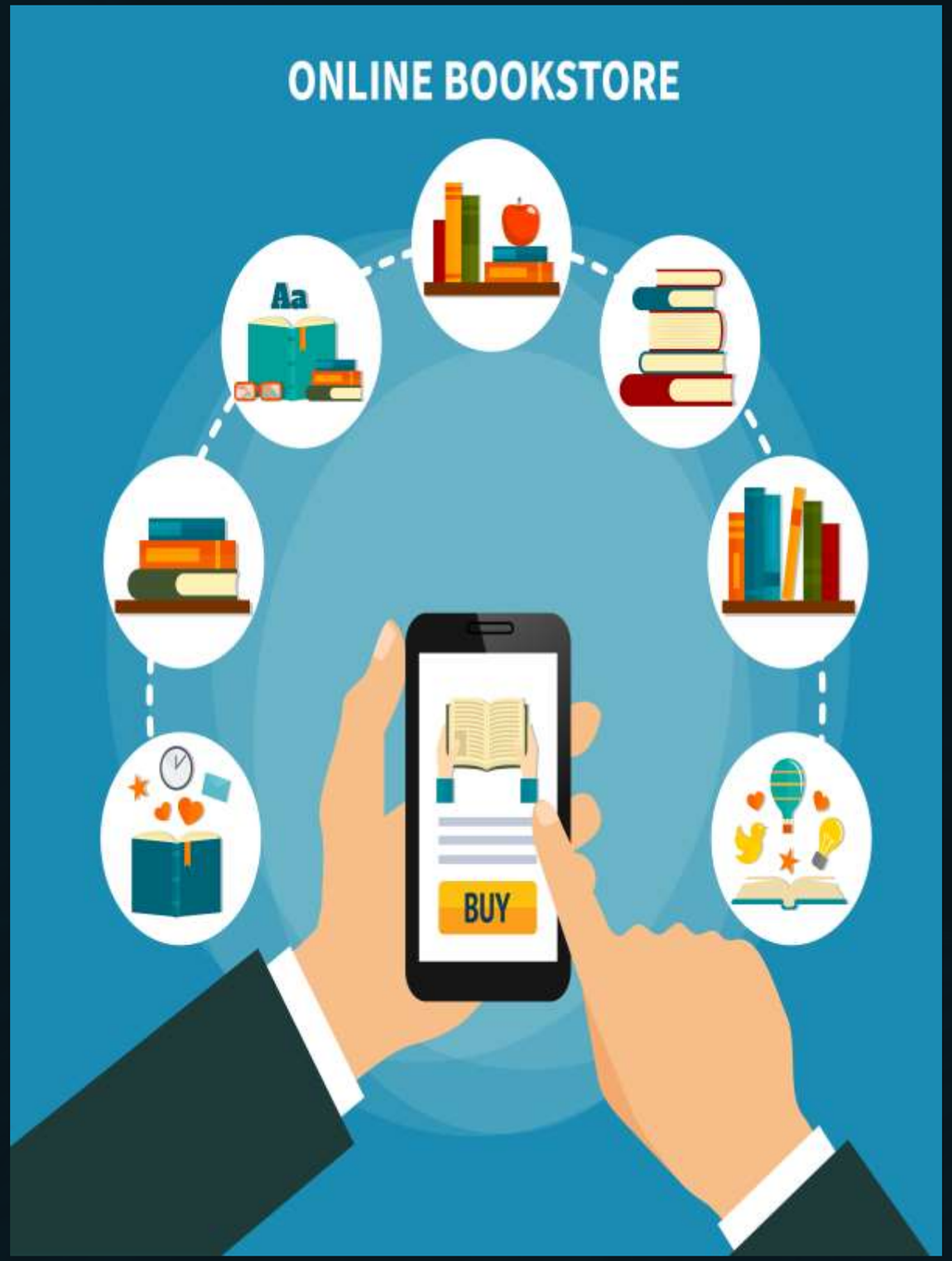
Online Bookstore Management System

The **Online Bookstore Management System** is a database project designed to manage the core functions of a digital bookshop using SQL. It focuses on storing and retrieving information about **customers**, **books**, and their **orders**. The system helps in organizing book data, tracking customer purchases, and simplifying order processing.

ENTITY RELATIONSHIP DIAGRAM

An **ER diagram** (Entity-Relationship diagram) in MySQL Workbench is a visual representation of the database structure. It shows how tables (entities) in the database are related to each other.



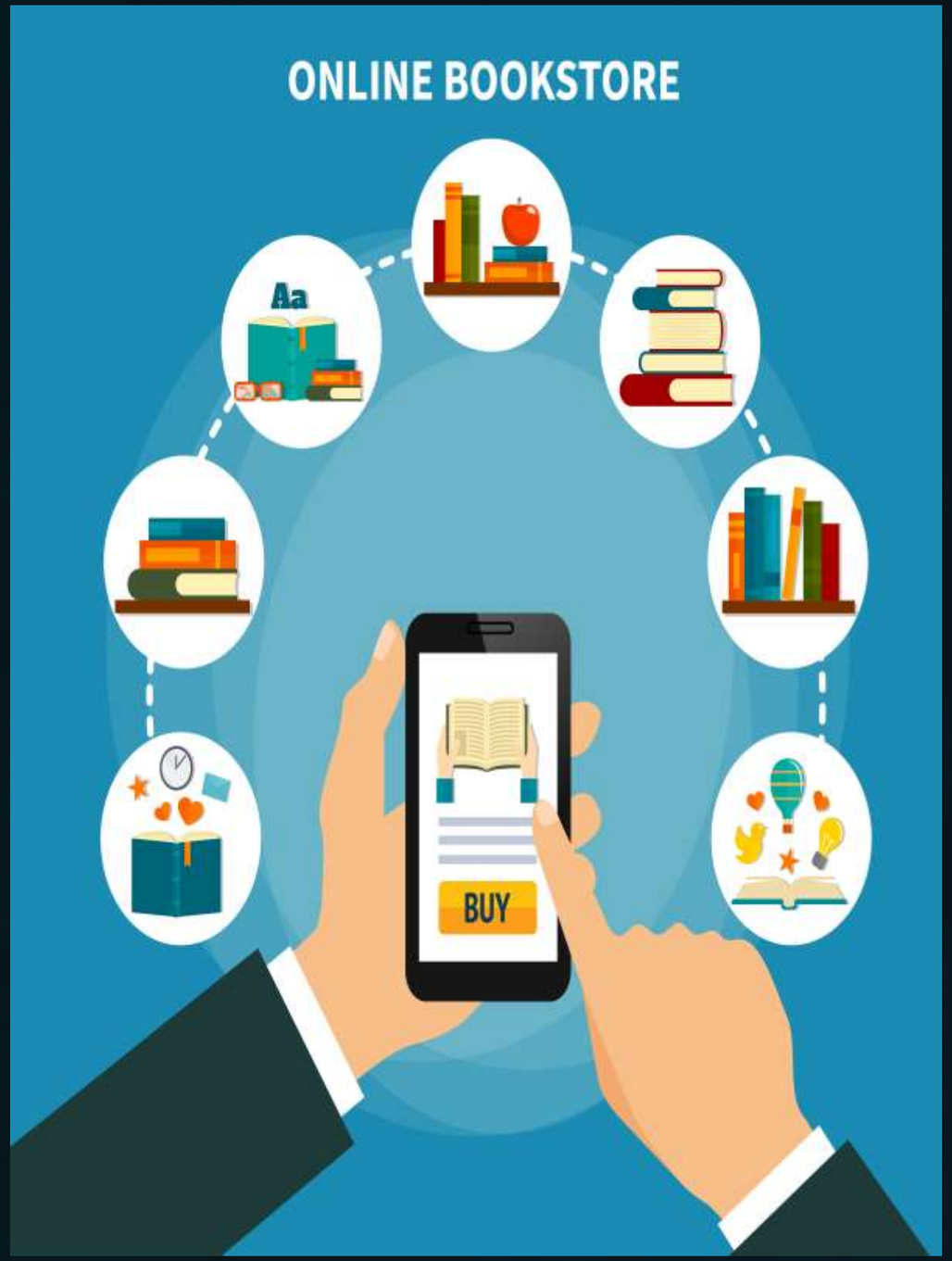


STRUCTURE OF TABLE

```
describe customers;
```

	Field	Type	Null	Key	Default	Extra
▶	CustomerID	int	NO	PRI	NULL	
	Name	varchar(100)	YES		NULL	
	Email	varchar(100)	YES		NULL	
	Phone	varchar(15)	YES		NULL	

It gives the structure of table customers that we have created

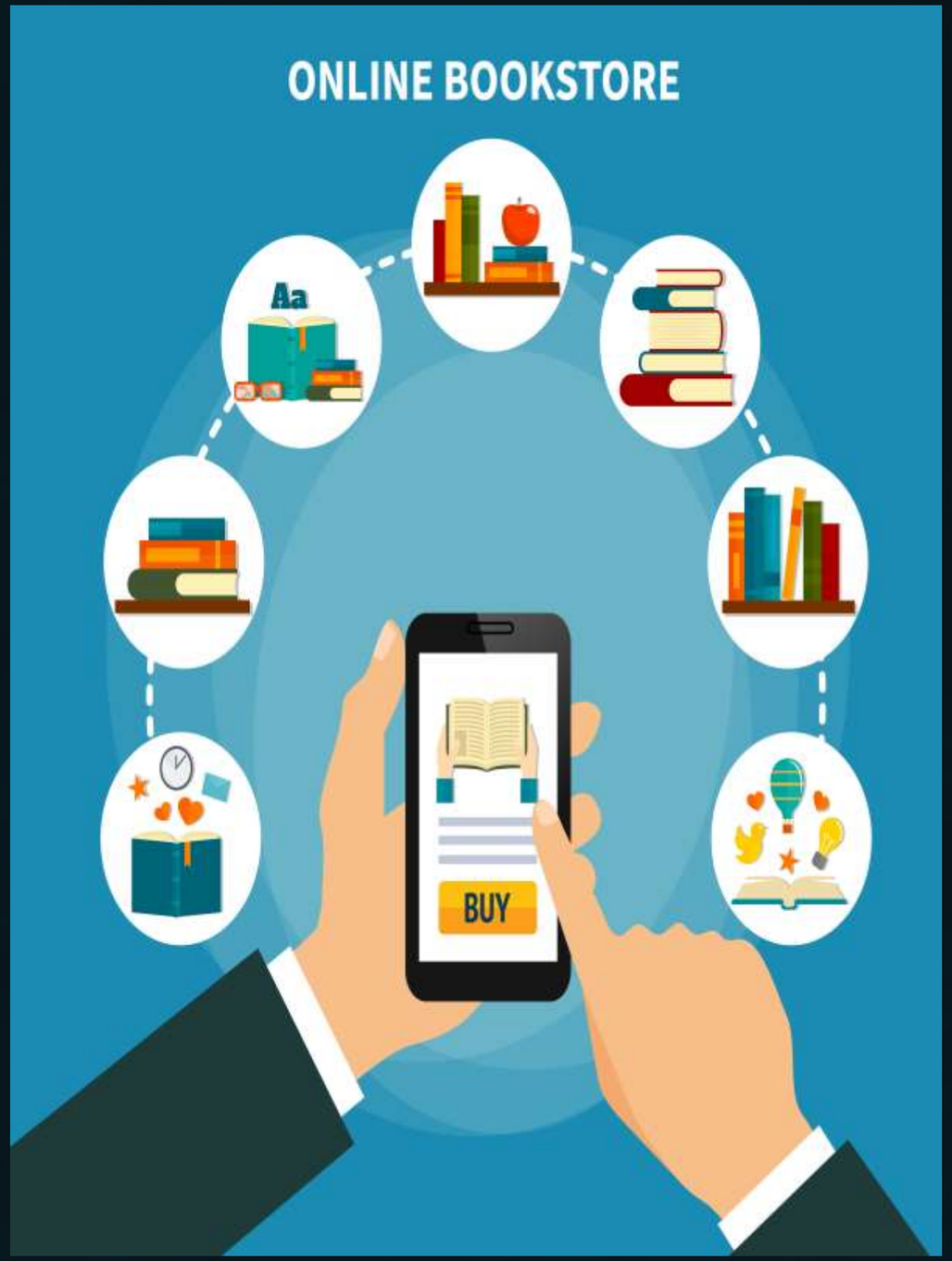


STRUCTURE OF TABLE

```
describe books;
```

	Field	Type	Null	Key	Default	Extra
▶	BookID	int	NO	PRI	NULL	
	Title	varchar(100)	YES		NULL	
	Author	varchar(100)	YES		NULL	
	Price	decimal(10,2)	YES		NULL	
	Stock	int	YES		NULL	

It gives the structure of table books that we have created



STRUCTURE OF TABLE

```
describe orders;
```

	Field	Type	Null	Key	Default	Extra
▶	OrderID	int	NO	PRI	NULL	
	CustomerID	int	YES	MUL	NULL	
	BookID	int	YES	MUL	NULL	
	Quantity	int	YES		NULL	
	OrderDate	date	YES		NULL	

It gives the structure of table orders that we have created

CONTENTS OF TABLE

```
select * from customers;
```

CustomerID	Name	Email	Phone
1	Alice Smith	alice@example.com	1234567890
2	Bob Johnson	bob@example.com	9876543210
3	Clara Kent	clara@example.com	5551234567
NULL	NULL	NULL	NULL

It gives the contents of table customers that we have created

CONTENTS OF TABLE

```
select * from orders;
```

	OrderID	CustomerID	BookID	Quantity	OrderDate
▶	201	1	101	2	2025-05-01
	202	2	103	1	2025-05-02
	203	1	102	3	2025-05-03
*	NULL	NULL	NULL	NULL	NULL

It gives the content of table orders that we have created

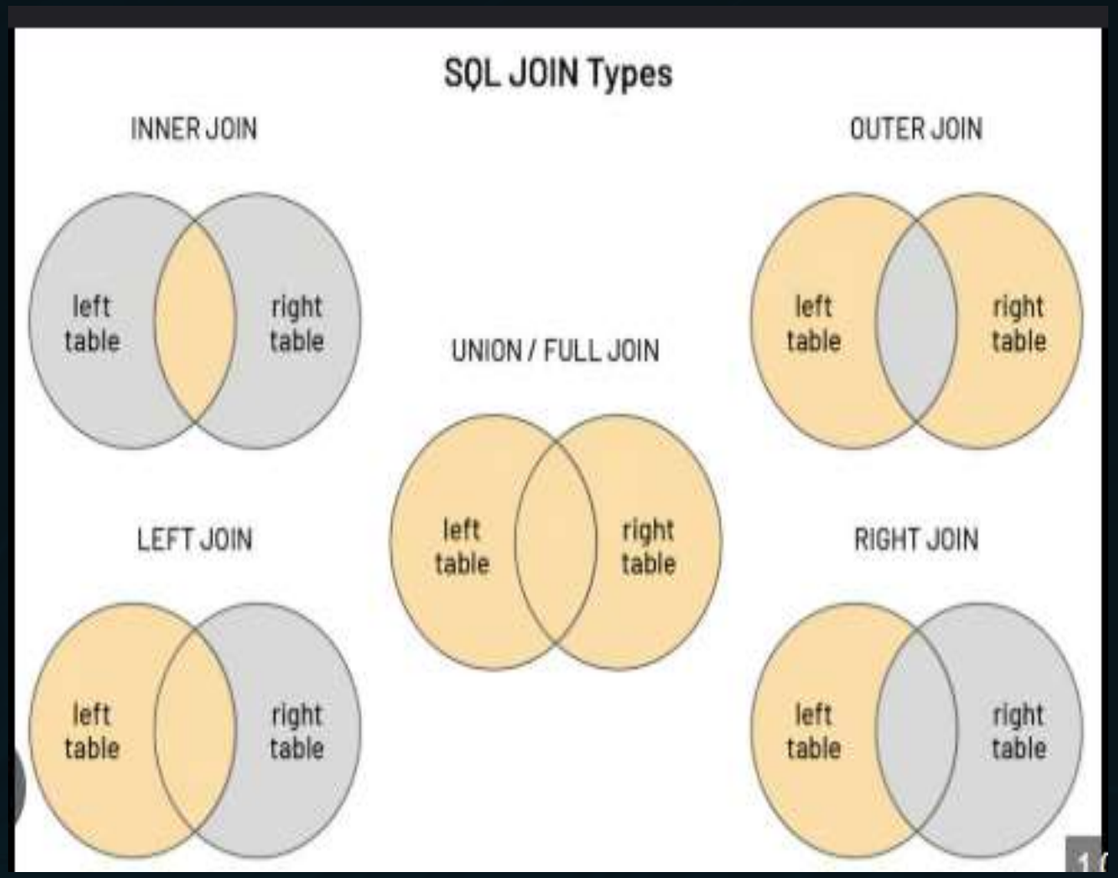
CONTENTS OF TABLE

```
select * from books;
```

BookID	Title	Author	Price	Stock
101	The Alchemist	Paulo Coelho	15.99	10
102	1984	George Orwell	12.49	5
103	Clean Code	Robert C. Martin	35.00	8
NULL	NULL	NULL	NULL	NULL

It gives the content of table books that we have created

JOINS



JOIN QUERIES

1. Show customer names and their order IDs

```sql
-- 1. Show customer names and their order IDs
SELECT Customers.Name, Orders.OrderID
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Name	OrderID
Alice Smith	201
Alice Smith	203
Bob Johnson	202

This query matches each customer with their orders.
It shows the **name of the customer** and the **ID of the order** they placed.

2. Show book titles and their order IDs

```sql
-- 2. Show book titles and their order IDs
SELECT Books.Title, Orders.OrderID
FROM Books
JOIN Orders ON Books.BookID = Orders.BookID;
```

Title	OrderID
The Alchemist	201
1984	203
Clean Code	202

This query matches each book with the orders that include it.
It shows the **title of the book** and the **ID of the order** that includes that book.

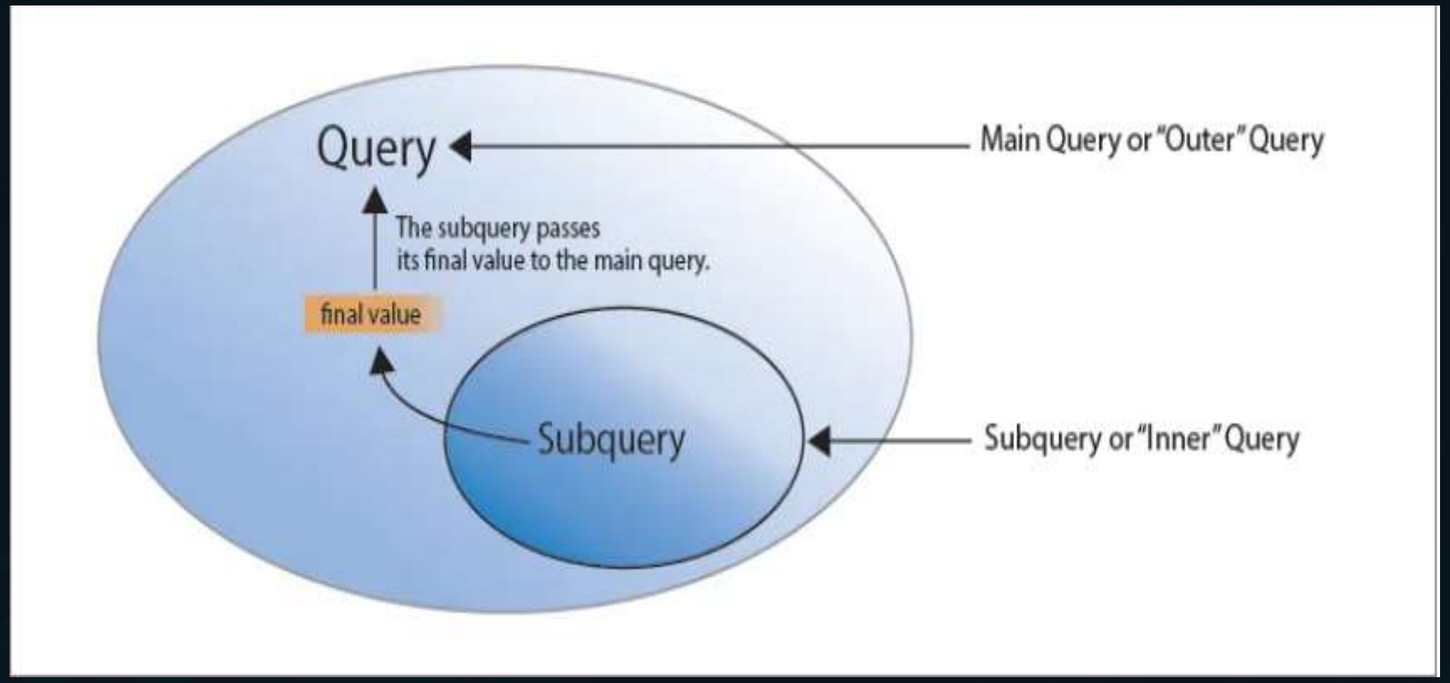
3. Show customer name & book title for each order

```sql
-- 3. Show customer name and book title for each order
SELECT Customers.Name, Books.Title
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN Books ON Orders.BookID = Books.BookID;
```

Name	Title
Alice Smith	The Alchemist
Alice Smith	1984
Bob Johnson	Clean Code

This query links **customers**, **orders**, and **books** together.
It shows which **customer ordered which book**.

SUB QUERY



SUB QUERIES

1.Books that cost more than avg price IDs

```
SELECT Title FROM Books
WHERE Price > (SELECT AVG(Price) FROM Books);
```

	Title
▶	Clean Code

The subquery select(avg(price) from books calculates average price for all books

2. Customers who have placed an order.s

```
-- 2. Customers who have placed an order
SELECT Name FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders);
```

	Name
▶	Alice Smith
	Bob Johnson

The subquery (select customer id from orders) gets a list of customer ids who placed an order

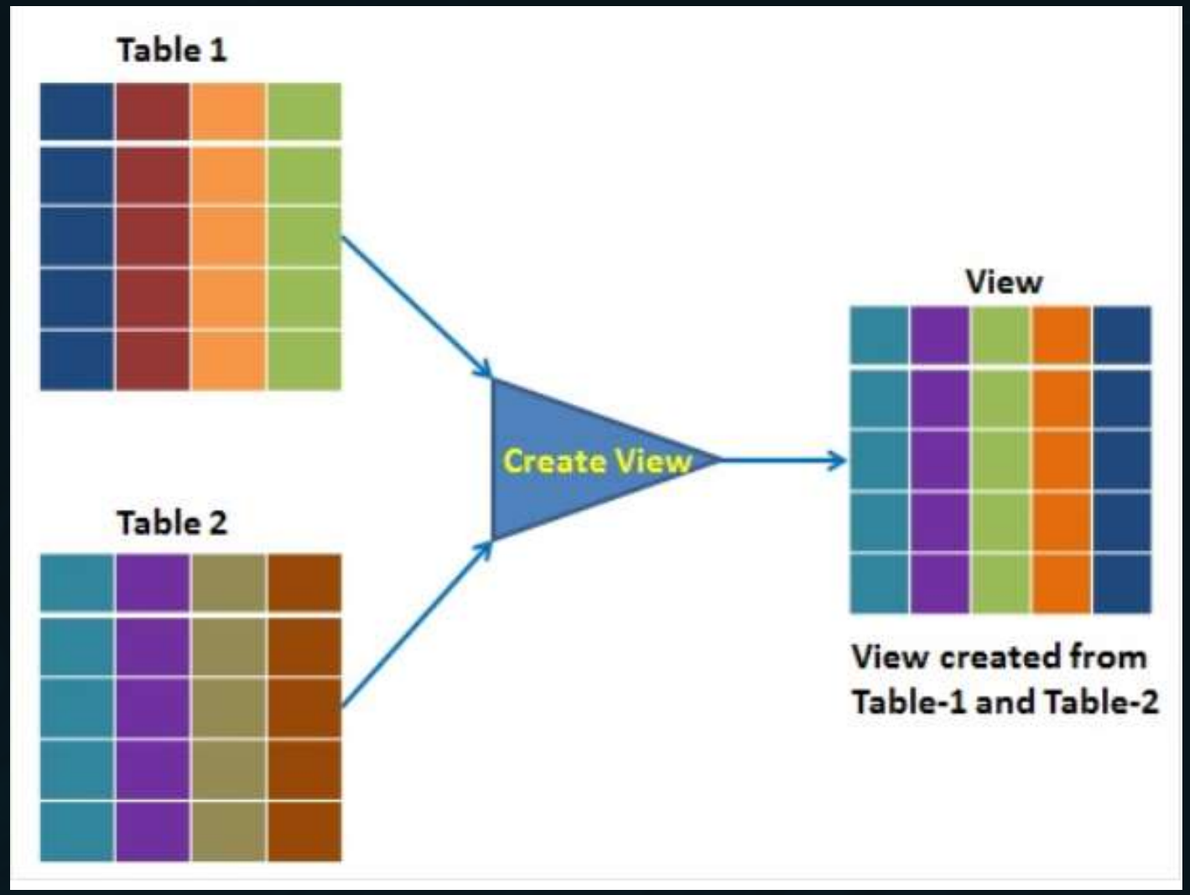
3. Titles of books that have been ordered

```
-- 3. Titles of books that have been ordered
SELECT Title FROM Books
WHERE BookID IN (SELECT BookID FROM Orders);
```

	Title
▶	The Alchemist
	1984
	Clean Code

The subquery (select book id from orders) fetches the list of book ids that were ordered

VIEWS



VIEW QUERIES

1. View with order and customer name.IDs

```sql
-- 1. View with order and customer name
CREATE VIEW SimpleOrderView AS
SELECT OrderID, Name
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
select * from SimpleOrderView;
```

OrderID	Name
201	Alice Smith
203	Alice Smith
202	Bob Johnson

Creates a view named (SimpleOrderView) that shows order IDs along with the names of the customers who placed them

2. View with book title and prices

```sql
-- 2. View with book title and price
CREATE VIEW BookPriceView AS
SELECT Title, Price FROM Books;
select * from BookPriceView;
```

Title	Price
The Alchemist	15.99
1984	12.49
Clean Code	35.00

Creates a view named (BookPriceView) that displays just the title and price of all books from the books table

3. View with customer and book they ordered

```sql
-- 3. View with customer and book they ordered
CREATE VIEW CustomerBookView AS
SELECT Customers.Name, Books.Title
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN Books ON Orders.BookID = Books.BookID;
select * from CustomerBookView;
```

Name	Title
Alice Smith	The Alchemist
Alice Smith	1984
Bob Johnson	Clean Code

Creates a view named (CustomerBookView) that shows which customer ordered which book by combining data from orders, customers and books

CONCLUSION

Through the creation of tables, data insertion, JOINs, subqueries, and views, we have built and interacted with a simple yet functional bookstore database. These SQL queries demonstrate how relational databases can be used to store, relate, and analyze data effectively. By linking customers, books, and orders:

- We explored how to combine data from multiple tables using JOINS.
- We learned to filter and extract meaningful insights using SUBQUERIES.
- We simplified complex logic into reusable views for easier data access.