

# Online Bookstore Database

Name: Debashmita Saha

Enrolment Number: EBEON0524136576

Batch: 2024-RDMS

## Problem Statement

This project shows the RDBMS usage to store and manipulate data of a bookstore that has information regarding Books in the store, Customers database, orders database, Authors' database, and Publishers data. This project showcases relations between the various components of the database. It uses data manipulation to find outcomes for complex calculations and maintain daily transaction records at the store.

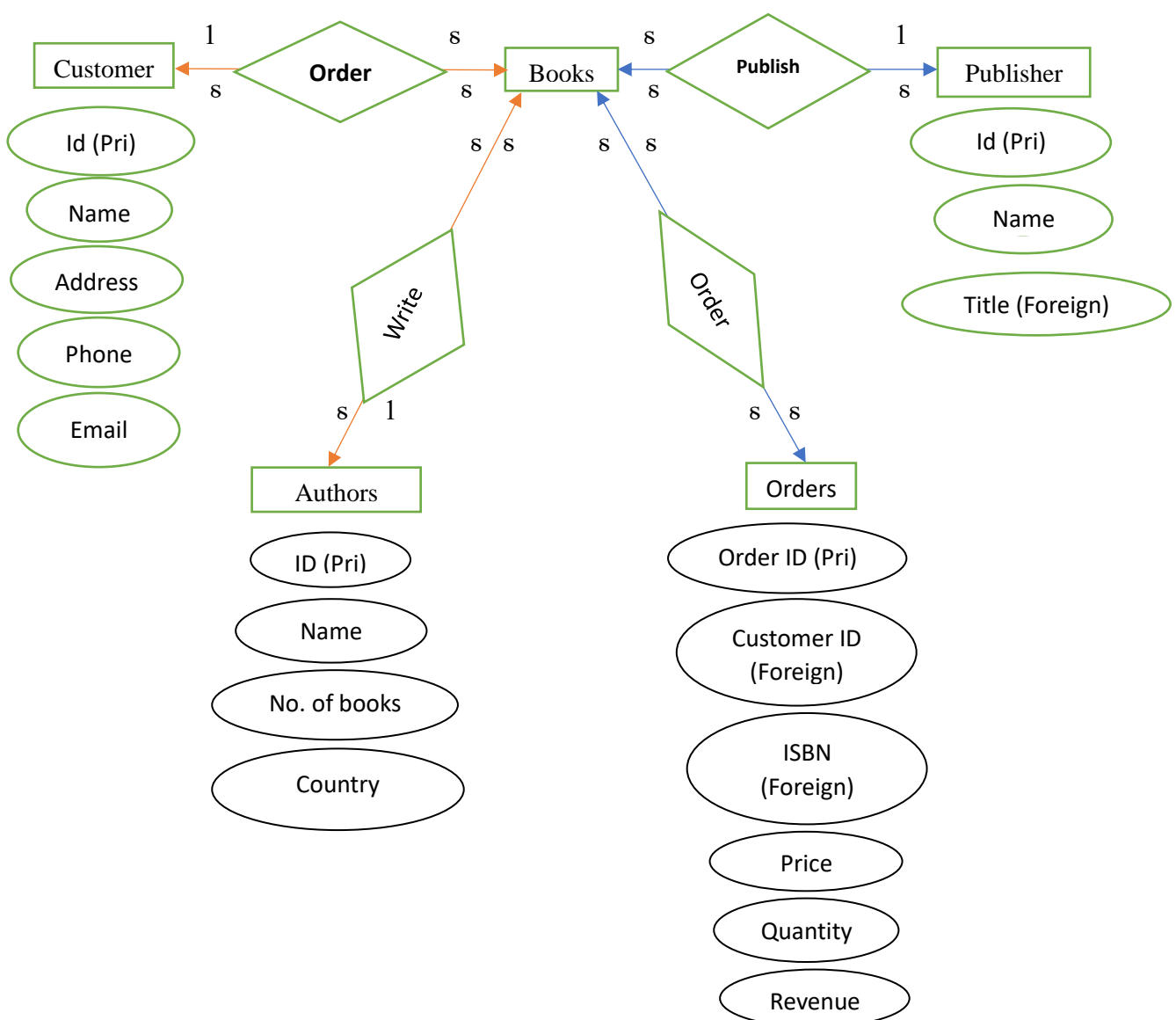
Books Table entities: Books identification code (SBIN), Title, Author, Publishing Date, Price, No. of copies, Rating, Genre.

Customer Table entities: Customer id, Name, Address, Phone number, Email.

Orders Table entities: Order id, Customer id, SBIN, Address, Price, Quantity, Total Amount.

Authors Table entities: Authors id, Name, Title, Number of Books Written, Rating, Country.

Publishers Table entities: ID, Name, Title, No. of copies publishes, Total number of books published.



## Assumption and Constraints

The code is tested on random dummy data and is therefore not accurate to the existing real-life information. The dataset created is solely for debugging and is not intended to derive any real-life analysis. In this project, the size of the dataset and available time to analyse has been a constant constraint.

## Codes and Output

### Part 1: Data Feeding

Creating the required tables with attributes and populating the data with random values.

```
create database Bookstore_Database; -- Database Creation
```

```
use Bookstore_Database;
```

```
create table Books_info
```

```
(
```

```
ISBN VARCHAR(100) PRIMARY KEY,
```

```
title VARCHAR(200) NOT NULL UNIQUE,
```

```
author VARCHAR(100) NOT NULL,
```

```
pub_date DATE,
```

```
genre VARCHAR(100),
```

```
price DECIMAL(10, 2) NOT NULL,
```

```
quant INT NOT NULL,
```

```
rating DECIMAL(10,2) NOT NULL
```

```
);
```

```
CREATE TABLE Customer_Info
```

```
(
```

```
cus_id INT PRIMARY KEY,
```

```
cus_name VARCHAR(100) NOT NULL,
```

```
phone VARCHAR(20) NOT NULL,
```

```
email VARCHAR(100),
```

```
address VARCHAR(500)
```

```
);
```

CREATE TABLE Authors

```
(
  Author_id VARCHAR(50) PRIMARY KEY,
  author VARCHAR(100) NOT NULL,
  No_of_books INT NOT NULL,
  Rating DECIMAL(10,2) NOT NULL,
  Country VARCHAR(20)
);
select * from Publishers;
```

CREATE TABLE Publishers

```
(
  Pub_id VARCHAR(50),
  Pub_name VARCHAR(100),
  title VARCHAR(200),
  FOREIGN KEY (title) REFERENCES Books_info(title)
);
```

CREATE TABLE Orders

```
(
  Order_id INT PRIMARY KEY,
  cus_id INT,
  ISBN VARCHAR(100),
  address VARCHAR(200),
  Price DECIMAL(10,2),
  Quant DECIMAL(10,2),
  Rev DECIMAL(10,2),
  FOREIGN KEY (ISBN) REFERENCES Books_info(ISBN),
```

FOREIGN KEY (cus\_id) REFERENCES Customer\_Info(cus\_id)

);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)  
VALUES

('ISBN00000001', 'Journey to the Center of the Earth', 'Jules Verne', '1864-11-25', 'Adventure',  
9.99, 100, 8.7),

('ISBN00000002', 'Pride and Prejudice', 'Jane Austen', '1813-01-28', 'Romance', 12.99, 120,  
9.1),

('ISBN00000003', 'To Kill a Mockingbird', 'Harper Lee', '1960-07-11', 'Drama', 8.99, 80, 9.3),

('ISBN00000004', '1984', 'George Orwell', '1949-06-08', 'Dystopian', 7.99, 200, 9.0),

('ISBN00000005', 'The Great Gatsby', 'F. Scott Fitzgerald', '1925-04-10', 'Classic', 14.99, 85,  
8.5),

('ISBN00000006', 'Moby Dick', 'Herman Melville', '1851-10-18', 'Epic', 11.99, 150, 8.1),

('ISBN00000007', 'War and Peace', 'Leo Tolstoy', '1869-01-01', 'Historical', 13.99, 70, 9.2),

('ISBN00000008', 'The Catcher in the Rye', 'J.D. Salinger', '1951-07-16', 'Literary Fiction',  
6.99, 90, 8.6),

('ISBN00000009', 'The Hobbit', 'J.R.R. Tolkien', '1937-09-21', 'Fantasy', 10.99, 110, 9.4),

('ISBN00000010', 'Frankenstein', 'Mary Shelley', '1818-01-01', 'Horror', 5.99, 60, 8.9);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)  
VALUES

('ISBN00000011', 'Brave New World', 'Aldous Huxley', '1932-01-01', 'Dystopian', 15.99, 65,  
8.3),

('ISBN00000012', 'The Picture of Dorian Gray', 'Oscar Wilde', '1890-06-20', 'Philosophical',  
10.99, 80, 8.6),

('ISBN00000013', 'The Brothers Karamazov', 'Fyodor Dostoevsky', '1880-11-01',  
'Philosophical', 17.99, 75, 9.0),

('ISBN00000014', 'Don Quixote', 'Miguel de Cervantes', '1605-01-16', 'Classic', 18.99, 60, 8.2),

('ISBN00000015', 'Invisible Man', 'Ralph Ellison', '1952-04-14', 'Literary Fiction', 16.99, 70,  
8.7),

('ISBN00000016', 'Gone with the Wind', 'Margaret Mitchell', '1936-06-30', 'Historical', 14.99,  
85, 8.8),

('ISBN00000017', 'Jane Eyre', 'Charlotte Brontë', '1847-10-16', 'Gothic', 12.99, 90, 8.9),

('ISBN00000018', 'The Grapes of Wrath', 'John Steinbeck', '1939-04-14', 'Realist', 13.99, 100,  
9.1),

('ISBN0000019', 'Wuthering Heights', 'Emily Brontë', '1847-12-01', 'Tragedy', 11.99, 95, 8.4),  
('ISBN0000020', 'The Adventures of Sherlock Holmes', 'Arthur Conan Doyle', '1892-10-14',  
'Mystery', 9.99, 120, 9.2);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)  
VALUES

('ISBN0000021', 'The Alchemist', 'Paulo Coelho', '1988-01-01', 'Philosophical Fiction', 17.00,  
150, 8.2),

('ISBN0000022', 'The Da Vinci Code', 'Dan Brown', '2003-03-18', 'Thriller', 18.00, 80, 7.5),

('ISBN0000023', 'Sapiens', 'Yuval Noah Harari', '2011-01-01', 'Non-fiction', 19.99, 70, 9.1),

('ISBN0000024', 'Life of Pi', 'Yann Martel', '2001-09-11', 'Adventure Fiction', 14.00, 100,  
8.0),

('ISBN0000025', 'The Road', 'Cormac McCarthy', '2006-09-26', 'Post-Apocalyptic', 15.00, 90,  
7.9),

('ISBN0000026', 'Eleanor Oliphant Is Completely Fine', 'Gail Honeyman', '2017-05-09',  
'Contemporary Fiction', 16.00, 85, 8.1),

('ISBN0000027', 'Educated', 'Tara Westover', '2018-02-20', 'Memoir', 20.00, 95, 9.0),

('ISBN0000028', 'The Night Circus', 'Erin Morgenstern', '2011-09-13', 'Fantasy', 12.00, 110,  
8.3),

('ISBN0000029', 'The Goldfinch', 'Donna Tartt', '2013-09-23', 'Literary Fiction', 22.00, 60,  
7.8),

('ISBN0000030', 'The Martian', 'Andy Weir', '2011-02-11', 'Science Fiction', 13.00, 120, 8.9);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)  
VALUES

('ISBN0000031', 'The Silent Patient', 'Alex Michaelides', '2019-02-05', 'Thriller', 24.99, 150,  
8.6),

('ISBN0000032', 'Where the Crawdads Sing', 'Delia Owens', '2018-08-14', 'Mystery', 18.99,  
80, 8.9),

('ISBN0000033', 'Becoming', 'Michelle Obama', '2018-11-13', 'Memoir', 22.99, 120, 9.2),

('ISBN0000034', 'Uneducated', 'Tara Westover', '2018-02-20', 'Memoir', 20.99, 95, 9.0),

('ISBN0000035', 'Normal People', 'Sally Rooney', '2018-08-28', 'Romance', 17.99, 85, 8.3),

('ISBN0000036', 'The Body Keeps the Score', 'Bessel van der Kolk', '2014-09-25',  
'Psychology', 16.99, 90, 9.1),

('ISBN0000037', 'Atomic Habits', 'James Clear', '2018-10-16', 'Self-help', 19.99, 70, 9.3),

('ISBN0000038', 'Dune', 'Frank Herbert', '1965-08-01', 'Science Fiction', 15.99, 110, 8.7),  
('ISBN0000039', 'The Four Agreements', 'Don Miguel Ruiz', '1997-11-07', 'Spirituality',  
12.99, 75, 8.4),  
('ISBN0000040', 'The Subtle Art of Not Giving a F\*ck', 'Mark Manson', '2016-09-13', 'Self-  
help', 21.99, 65, 8.5);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN0000041', 'The Namesake', 'Jhumpa Lahiri', '2003-01-01', 'Fiction', 19.99, 100, 4.5),  
('ISBN0000042', 'The God of Small Things', 'Arundhati Roy', '1997-05-01', 'Literary  
Fiction', 24.99, 80, 4.8),  
('ISBN0000043', 'A Suitable Boy', 'Vikram Seth', '1993-11-01', 'Historical Fiction', 29.99,  
120, 4.7),  
('ISBN0000044', 'The White Tiger', 'Aravind Adiga', '2008-03-01', 'Contemporary Fiction',  
21.99, 150, 4.6),  
('ISBN0000045', 'The Palace of Illusions', 'Chitra Banerjee Divakaruni', '2008-02-01',  
'Mythological Fiction', 18.99, 90, 4.4),  
('ISBN0000046', 'The Inheritance of Loss', 'Kiran Desai', '2006-09-01', 'Literary Fiction',  
23.99, 110, 4.7),  
('ISBN0000047', 'The Shadow Lines', 'Amitav Ghosh', '1988-01-01', 'Historical Fiction',  
20.99, 70, 4.3),  
('ISBN0000048', 'Midnight's Children', 'Salman Rushdie', '1981-09-01', 'Magical Realism',  
26.99, 130, 4.9),  
('ISBN0000049', 'Train to Pakistan', 'Khushwant Singh', '1956-01-01', 'Historical Fiction',  
17.99, 60, 4.2),  
('ISBN0000050', 'The Guide', 'R.K. Narayan', '1958-07-01', 'Classic Fiction', 22.99, 100,  
4.5);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN0000051', 'Brida', 'Paulo Coelho', '1990-03-15', 'Magical Realism', 12.99, 150, 4.5),  
('ISBN0000052', 'Veronika Decides to Die', 'Paulo Coelho', '1998-05-20', 'Psychological  
Fiction', 16.99, 180, 4.6);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN00000053', 'Angels & Demons', 'Dan Brown', '2000-03-28', 'Thriller', 15.99, 120, 4.6),  
('ISBN00000054', 'Inferno', 'Dan Brown', '2013-05-14', 'Mystery', 18.99, 90, 4.4),  
('ISBN00000055', 'Origin', 'Dan Brown', '2017-10-03', 'Adventure', 21.99, 150, 4.8);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN00000056', 'The Art of Noticing', 'James Clear', '2019-07-15', 'Self-Help', 14.99, 80, 4.5),  
('ISBN00000057', 'Make Time', 'James Clear', '2018-09-20', 'Productivity', 12.99, 100, 4.3),  
('ISBN00000058', 'The Power of Habit', 'James Clear', '2012-02-28', 'Psychology', 16.99, 120, 4.7);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN00000059', 'Interpreter of Maladies', 'Jhumpa Lahiri', '1999-06-01', 'Short Stories', 17.99, 90, 4.6),  
('ISBN00000060', 'Unaccustomed Earth', 'Jhumpa Lahiri', '2008-04-01', 'Fiction', 19.99, 110, 4.7),  
('ISBN00000061', 'The Lowland', 'Jhumpa Lahiri', '2013-09-01', 'Historical Fiction', 21.99, 120, 4.5);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN00000062', 'Malgudi Days', 'R.K. Narayan', '1943-07-01', 'Short Stories', 14.99, 80, 4.5),  
('ISBN00000063', 'The Bachelor of Arts', 'R.K. Narayan', '1937-11-15', 'Literary Fiction', 12.99, 100, 4.3),  
('ISBN00000064', 'The English Teacher', 'R.K. Narayan', '1945-03-20', 'Novel', 16.99, 120, 4.7);

INSERT INTO Books\_info (ISBN, title, author, pub\_date, genre, price, quant, rating)

VALUES

('ISBN00000065', 'The Maidens', 'Alex Michaelides', '2021-06-01', 'Psychological Thriller', 14.99, 100, 4.5),  
('ISBN00000066', 'The Fury', 'Alex Michaelides', '2024-03-15', 'Mystery', 16.99, 120, 4.7);



```
select * from books_info;
```

```
desc books_info;
```

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	Field	Type	Null	Key	Default	Extra
►	ISBN	varchar(100)	NO	PRI	NULL	
	title	varchar(200)	NO	UNI	NULL	
	author	varchar(100)	NO		NULL	
	pub_date	date	YES		NULL	
	genre	varchar(100)	YES		NULL	
	price	decimal(10,2)	NO		NULL	
	quant	int	NO		NULL	
	rating	decimal(10,2)	NO		NULL	

Fig 1 Showing the structure of Books\_info table.

```
INSERT INTO Customer_Info (cus_id, cus_name, phone, email, address)
```

```
VALUES
```

```
(101, 'John Doe', '+1 123-456-789', 'john.doe@gmail.com', '123 Main St, Anytown, USA'),
```

```
(102, 'Jane Smith', '+44 20 1234 5678', 'jane.smith@gmail.co.uk', '456 Elm Ave, London, UK'),
```

```
(103, 'Alice Johnson', '+81 3 1234 5678', 'alice.johnson@gmail.jp', '789 Sakura St, Tokyo, Japan'),
```

```
(104, 'Robert Lee', '+49 30 1234 5678', 'robert.lee@gmail.de', '987 Linden Strasse, Berlin, Germany'),
```

```
(105, 'Maria Rodriguez', '+34 91 123 4567', 'maria.rodriguez@gmail.es', '654 Calle Real, Madrid, Spain'),
```

```
(106, 'David Kim', '+82 2 1234 5678', 'david.kim@gmail.kr', '321 Gwanghwamun St, Seoul, South Korea'),
```

```
(107, 'Emily Chen', '+86 10 1234 5678', 'emily.chen@gmail.cn', '987 Tiananmen Rd, Beijing, China'),
```

```
(108, 'Michael Nguyen', '+84 24 1234 5678', 'michael.nguyen@gmail.vn', '456 Nguyen Hue St, Ho Chi Minh City, Vietnam'),
```

```
(109, 'Sophia Garcia', '+52 55 1234 5678', 'sophia.garcia@gmail.mx', '789 Reforma Ave, Mexico City, Mexico'),
```

(110, 'William Brown', '+61 2 1234 5678', 'william.brown@gmail.au', '123 George St, Sydney, Australia'),

(111, 'Olivia Patel', '+91 22 1234 5678', 'olivia.patel@gmail.in', '654 MG Road, Mumbai, India'),

(112, 'Ethan Nguyen', '+84 28 1234 5678', 'ethan.nguyen@gmail.vn', '987 Le Loi St, Ho Chi Minh City, Vietnam'),

(113, 'Ava Kim', '+82 51 1234 5678', 'ava.kim@gmail.kr', '321 Haeundae Beach Rd, Busan, South Korea'),

(114, 'Liam Wang', '+86 21 1234 5678', 'liam.wang@gmail.cn', '456 Nanjing Rd, Shanghai, China'),

(115, 'Isabella Tan', '+65 6 1234 5678', 'isabella.tan@gmail.sg', '789 Orchard Rd, Singapore');

INSERT INTO Customer\_Info (cus\_id, cus\_name, phone, email, address)

VALUES

(116, 'Henry Adams', '+1 987-654-321', 'henry.adams@gmail.com', '246 Oak Lane, Springfield, USA'),

(117, 'Chloe Brown', '+44 20 9876 5432', 'chloe.brown@gmail.co.uk', '789 Maple Rd, Manchester, UK'),

(118, 'Liam Chen', '+81 3 9876 5432', 'liam.chen@gmail.jp', '123 Sakura St, Kyoto, Japan'),

(119, 'Emma Davis', '+49 30 9876 5432', 'emma.davis@gmail.de', '456 Linden Strasse, Hamburg, Germany'),

(120, 'Ethan Evans', '+34 91 987 6543', 'ethan.evans@gmail.es', '654 Calle Real, Barcelona, Spain'),

(121, 'Ava Foster', '+82 2 9876 5432', 'ava.foster@example.kr', '321 Gwanghwamun St, Seoul, South Korea'),

(122, 'Oliver Garcia', '+86 10 9876 5432', 'oliver.garcia@gmail.cn', '987 Tiananmen Rd, Shanghai, China'),

(123, 'Mia Hernandez', '+84 24 9876 5432', 'mia.hernandez@gmail.vn', '456 Nguyen Hue St, Hanoi, Vietnam'),

(124, 'Noah Jackson', '+52 55 9876 5432', 'noah.jackson@gmail.mx', '789 Reforma Ave, Guadalajara, Mexico'),

(125, 'Sophia Kim', '+61 2 9876 5432', 'sophia.kim@example.au', '123 George St, Melbourne, Australia'),

(126, 'Lucas Lee', '+91 22 9876 5432', 'lucas.lee@gmail.in', '654 MG Road, Bangalore, India'),

(127, 'Olivia Martinez', '+84 28 9876 5432', 'olivia.martinez@gmail.vn', '987 Le Loi St, Da Nang, Vietnam'),

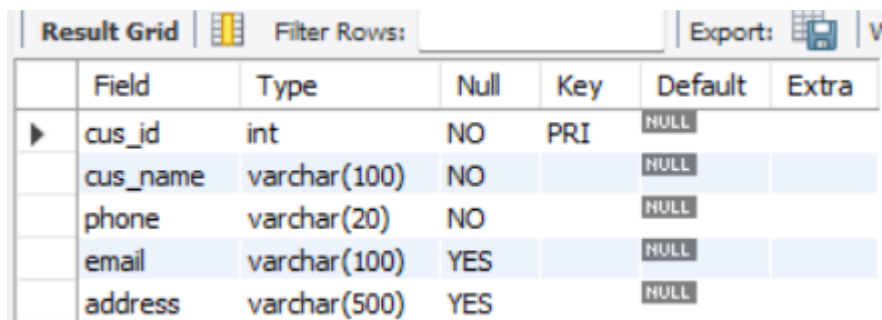
(128, 'Mason Nguyen', '+82 51 9876 5432', 'mason.nguyen@gmail.kr', '321 Haeundae Beach Rd, Busan, South Korea'),

(129, 'Isabella Patel', '+86 21 9876 5432', 'isabella.patel@gmail.cn', '456 Nanjing Rd, Hangzhou, China'),

(130, 'Alexander Tan', '+65 6 9876 5432', 'alexander.tan@gmail.sg', '789 Orchard Rd, Singapore');

```
select * from Customer_Info;
```

```
desc Customer_Info;
```



	Field	Type	Null	Key	Default	Extra
▶	cus_id	int	NO	PRI	NULL	
	cus_name	varchar(100)	NO		NULL	
	phone	varchar(20)	NO		NULL	
	email	varchar(100)	YES		NULL	
	address	varchar(500)	YES		NULL	

Fig 2 : Showing the structure of Customer's table

```
INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES
```

('AUT00001', 'Jules Verne', 1, 8.7, 'France'),

('AUT00002', 'Jane Austen', 1, 9.1, 'United Kingdom'),

('AUT00003', 'Harper Lee', 1, 9.3, 'United States'),

('AUT000014', 'George Orwell', 1, 9.0, 'United Kingdom'),

('AUT00005', 'F. Scott Fitzgerald', 1, 8.5, 'United States'),

('AUT00006', 'Herman Melville', 1, 8.1, 'United States'),

('AUT00007', 'Leo Tolstoy', 1, 9.2, 'Russia'),

('AUT00008', 'J.D. Salinger', 1, 8.6, 'United States'),

('AUT00009', 'J.R.R. Tolkien', 1, 9.4, 'United Kingdom'),

('AUT00010', 'Mary Shelley', 1, 8.9, 'United Kingdom');

```

INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES
('AUT00011', 'Aldous Huxley', 1, 8.3, 'United Kingdom'),
('AUT00012', 'Oscar Wilde', 1, 8.6, 'Ireland'),
('AUT00013', 'Fyodor Dostoevsky', 1, 9.0, 'Russia'),
('AUT00014', 'Miguel de Cervantes', 1, 8.2, 'Spain'),
('AUT00015', 'Ralph Ellison', 1, 8.7, 'United States'),
('AUT00016', 'Margaret Mitchell', 1, 8.8, 'United States'),
('AUT00017', 'Charlotte Brontë', 1, 8.9, 'United Kingdom'),
('AUT00018', 'John Steinbeck', 1, 9.1, 'United States'),
('AUT00019', 'Emily Brontë', 1, 8.4, 'United Kingdom'),
('AUT00020', 'Arthur Conan Doyle', 1, 9.2, 'United Kingdom');

```

```

INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES
('AUT00021', 'Paulo Coelho', 3, 8.2, 'Brazil'),
('AUT00022', 'Dan Brown', 4, 7.5, 'United States'),
('AUT00023', 'Yuval Noah Harari', 1, 9.1, 'Israel'),
('AUT00024', 'Yann Martel', 1, 8.0, 'Canada'),
('AUT00025', 'Cormac McCarthy', 1, 7.9, 'United States'),
('AUT00026', 'Gail Honeyman', 1, 8.1, 'United Kingdom'),
('AUT00027', 'Tara Westover', 1, 9.0, 'United States'),
('AUT00028', 'Erin Morgenstern', 1, 8.3, 'United States'),
('AUT00029', 'Donna Tartt', 1, 7.8, 'United States'),
('AUT00030', 'Andy Weir', 1, 8.9, 'United States');

```

```

INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES
('AUT00031', 'Alex Michaelides', 3, 8.6, 'United Kingdom'),
('AUT00032', 'Delia Owens', 1, 8.9, 'United States'),

```

```

('AUT00033', 'Michelle Obama', 1, 9.2, 'United States'),
('AUT00034', 'Tara Westover', 1, 9.0, 'United States'),
('AUT00035', 'Sally Rooney', 1, 8.3, 'Ireland'),
('AUT00036', 'Bessel van der Kolk', 1, 9.1, 'Netherlands'),
('AUT00037', 'James Clear', 4, 9.3, 'United States'),
('AUT00038', 'Frank Herbert', 1, 8.7, 'United States'),
('AUT00039', 'Don Miguel Ruiz', 1, 8.4, 'Mexico'),
('AUT00040', 'Mark Manson', 1, 8.5, 'United States');

```

```

INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES

```

```

('AUT00041', 'Jhumpa Lahiri', 4, 7.3, 'India'),
('AUT00042', 'R.K. Narayan', 4, 8.6, 'India');

```

```

INSERT INTO Authors (Author_id, author, No_of_books, Rating, Country)
VALUES

```

```

('AUT00043', 'Vikram Seth', 1, 6.7, 'India'),
('AUT00044', 'Arundhati Roy', 1, 6.9, 'India'),
('AUT00045', 'Vikram Seth', 1, 7.1, 'India'),
('AUT00046', 'Aravind Adiga', 1, 7.0, 'India'),
('AUT00047', 'Chitra Banerjee Divakaruni', 1, 7.6, 'India'),
('AUT00048', 'Kiran Desai', 1, 7.8, 'India'),
('AUT00049', 'Amitav Ghosh', 1, 7.7, 'India'),
('AUT00050', 'Salman Rushdie', 1, 7.8, 'India'),
('AUT00051', 'Khushwant Singh', 1, 8.2, 'India');

```

```
DESC Authors;
```

Result Grid						
Filter Rows:						
	Field	Type	Null	Key	Default	Extra
►	Author_id	varchar(50)	NO	PRI	NULL	
	author	varchar(100)	NO		NULL	
	No_of_books	int	NO		NULL	
	Rating	decimal(10,2)	NO		NULL	
	Country	varchar(20)	YES		NULL	

Fig 3: Showing the structure of Author's table.

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES ('PUB001', 'Secker & Warburg', 'Journey to the Center of the Earth'),
('PUB002', 'J. B. Lippincott & Co.', 'Pride and Prejudice'),
('PUB003', 'J. B. Lippincott & Co.', 'To Kill a Mockingbird'),
('PUB004', 'Secker & Warburg', '1984'),
('PUB005', 'Secker & Warburg', 'The Great Gatsby'),
('PUB006', 'Secker & Warburg', 'Moby Dick'),
('PUB007', 'Secker & Warburg', 'War and Peace'),
('PUB008', 'Secker & Warburg', 'The Catcher in the Rye'),
('PUB009', 'Secker & Warburg', 'The Hobbit'),
('PUB010', 'Secker & Warburg', 'Frankenstein');
```

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
('PUB011', 'Chatto & Windus', 'Brave New World'),
('PUB012', 'WardLock', 'The Picture of Dorian Gray'),
('PUB013', 'The Russian Messenger', 'The Brothers Karamazov'),
('PUB014', 'Francisco de Robles', 'Don Quixote'),
('PUB015', 'Random House', 'Invisible Man');
```

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
('PUB016', 'Macmillan Publishers', 'Gone with the Wind'),
('PUB017', 'Smith, Elder and Co.', 'Jane Eyre'),
('PUB018', 'The Viking Press', 'The Grapes of Wrath'),
('PUB019', 'Thomas Cautley Newby', 'Wuthering Heights'),
('PUB020', 'George Newnes', 'The Adventures of Sherlock Holmes');
```

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB021', 'Penguin Books', 'The Alchemist'),

('PUB022', 'Doubleday', 'The Da Vinci Code'),

('PUB023', 'Random House', 'Sapiens'),

('PUB024', 'Knopf', 'Life of Pi'),

('PUB025', 'Vintage', 'The Road');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB026', 'Penguin Books', 'Eleanor Oliphant Is Completely Fine');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB027', 'Random House', 'Educated'),

('PUB028', 'Vintage', 'The Night Circus'),

('PUB029', 'Little, Brown and Company', 'The Goldfinch'),

('PUB030', 'Crown Publishing Group', 'The Martian');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB031', 'Celadon Books', 'The Silent Patient'),

('PUB032', 'Viking Press', 'Where the Crawdads Sing'),

('PUB033', 'Crown', 'Becoming'),

('PUB034', 'Random House', 'Uneducated'),

('PUB035', 'Faber & Faber', 'Normal People');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB036', 'HarperCollins', 'The Subtle Art of Not Giving a F\*ck'),

('PUB037', 'Penguin Books Ltd', 'Dune'),  
('PUB038', 'Amber-Allen Publishing', 'The Four Agreements'),  
('PUB039', 'Random House', 'The Body Keeps the Score'),  
('PUB040', 'Penguin Books Ltd', 'Atomic Habits');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)  
VALUES

('PUB041', 'HarperCollins', 'The Namesake'),  
('PUB042', 'Penguin Books Ltd', 'The God of Small Things'),  
('PUB043', 'Penguin Books Ltd', 'A Suitable Boy'),  
('PUB044', 'HarperCollins', 'The White Tiger'),  
('PUB045', 'Pan Macmillan India', 'The Palace of Illusions');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)  
VALUES

('PUB046', 'Atlantic Monthly Press', 'The Inheritance of Loss'),  
('PUB047', 'Hamish Hamilton', 'The Shadow Lines'),  
('PUB048', 'Viking Press', 'Midnight\'s Children'),  
('PUB049', 'Chatto & Windus', 'Train to Pakistan'),  
('PUB050', 'Methuen', 'The Guide');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)

VALUES

('PUB051', 'HarperCollins', 'Brida'),  
('PUB052', 'Penguin Books Ltd', 'Veronika Decides to Die');

INSERT INTO Publishers (Pub\_id, Pub\_name, title)  
VALUES

('PUB053', 'Random House', 'Angels & Demons'),  
('PUB054', 'Doubleday Publishing', 'Inferno'),  
('PUB055', 'Doubleday Publishing', 'Origin');



```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
```

```
('PUB056', 'Random House', 'The Art of Noticing'),
('PUB057', 'Penguin Books Ltd', 'Make Time'),
('PUB058', 'HarperCollins', 'The Power of Habit');
```

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
```

```
('PUB059', 'Knopf Doubleday Publishing Group', 'Interpreter of Maladies'),
('PUB060', 'Bloomsbury Press', 'Unaccustomed Earth'),
('PUB061', 'Random House', 'The Lowland');
```

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
```

```
('IndianThought', 'Indian Thought Publications', 'Malgudi Days'),
('PUB062', 'Nelson', 'The Bachelor of Arts'),
('PUB063', 'Eyre & Spottiswoode', 'The English Teacher');
```

```
INSERT INTO Publishers (Pub_id, Pub_name, title)
VALUES
```

```
('PUB064', 'Celadon Books', 'The Maidens'),
('PUB065', 'Celadon Books', 'The Fury');
```

```
select * from Publishers;
```

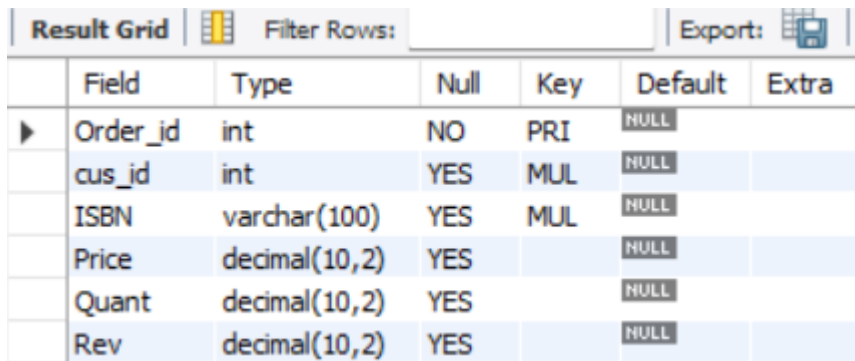
Result Grid						
Filter Rows:						
Export:						
	Field	Type	Null	Key	Default	Extra
▶	Pub_id	varchar(50)	YES		HULL	
	Pub_name	varchar(100)	YES		HULL	
	title	varchar(200)	YES	MUL	HULL	

Fig 4: Showing the structure of Publisher's Table

```
INSERT INTO Orders
VALUES (10001, 104, 'ISBN00000002', 9.99, 5, Price * Quant);
INSERT INTO Orders
VALUES(10002, 118,'ISBN00000047', 20.99, 8, Price * Quant);
INSERT INTO Orders
VALUES(10003, 101, 'ISBN00000002', 12.99, 6, Price * Quant);
INSERT INTO Orders
VALUES(10004, 102, 'ISBN00000034', 20.99, 3, Price * Quant);
INSERT INTO Orders
VALUES(10005, 103, 'ISBN00000008', 6.99, 2, Price * Quant);
INSERT INTO Orders
VALUES(10006, 103, 'ISBN00000015', 16.99, 1, Price * Quant);
INSERT INTO Orders
VALUES(10007, 102, 'ISBN00000041', 19.99, 2, Price * Quant);
INSERT INTO Orders
VALUES(10008, 104, 'ISBN00000041', 19.99, 1, Price * Quant);
INSERT INTO Orders
VALUES(10009, 106, 'ISBN00000042', 24.99, 1, Price * Quant);
INSERT INTO Orders
VALUES(10010, 106, 'ISBN00000017', 12.99, 12, Price * Quant);
INSERT INTO Orders
VALUES(10011, 107, 'ISBN00000044', 21.99, 3, Price * Quant);
INSERT INTO Orders
VALUES(10012, 108, 'ISBN00000019', 11.99, 10, Price * Quant);
INSERT INTO Orders
VALUES(10013, 109, 'ISBN00000009', 10.99, 5, Price * Quant);
INSERT INTO Orders
VALUES(10014, 110, 'ISBN00000010', 5.99, 12, Price * Quant);
INSERT INTO Orders
VALUES(10015, 110, 'ISBN00000044', 21.99, 11, Price * Quant);
```

```
INSERT INTO Orders
VALUES(10016, 111, 'ISBN0000019', 11.99, 20, Price * Quant);
INSERT INTO Orders
VALUES(10017, 112, 'ISBN0000053', 1, 15.99, Price * Quant);
INSERT INTO Orders
VALUES(10018, 113, 'ISBN0000053', 1, 15.99, Price * Quant);
INSERT INTO Orders
VALUES(10019, 113, 'ISBN0000038', 1, 15.99, Price * Quant);
INSERT INTO Orders
VALUES(10020, 113, 'ISBN0000033', 1, 22.99, Price * Quant);
INSERT INTO Orders
VALUES(10021, 113, 'ISBN0000032', 1, 18.99, Price * Quant);
INSERT INTO Orders
VALUES(10022, 113, 'ISBN0000034', 2, 20.99, Price * Quant);
INSERT INTO Orders
VALUES(10023, 114, 'ISBN0000066', 1, 16.99, Price * Quant);
INSERT INTO Orders
VALUES(10024, 110, 'ISBN0000037', 2, 19.99, Price * Quant);
INSERT INTO Orders
VALUES(10025, 121, 'ISBN0000021', 3, 17.00, Price * Quant);
INSERT INTO Orders
VALUES(10026, 122, 'ISBN0000022', 5, 18.00, Price * Quant);
INSERT INTO Orders
VALUES(10027, 111, 'ISBN0000024', 2, 14, Price * Quant);
INSERT INTO Orders
VALUES(10028, 112, 'ISBN0000025', 2, 15.00, Price * Quant);
INSERT INTO Orders
VALUES(10029, 115, 'ISBN0000059', 6, 17.99, Price * Quant);
INSERT INTO Orders
VALUES(10030, 120, 'ISBN0000060', 8, 19.99, Price * Quant);
```

```
select * from Orders;
```



	Field	Type	Null	Key	Default	Extra
►	Order_id	int	NO	PRI	NULL	
	cus_id	int	YES	MUL	NULL	
	ISBN	varchar(100)	YES	MUL	NULL	
	Price	decimal(10,2)	YES		NULL	
	Quant	decimal(10,2)	YES		NULL	
	Rev	decimal(10,2)	YES		NULL	

Fig 5: Showing the structure of Order's table

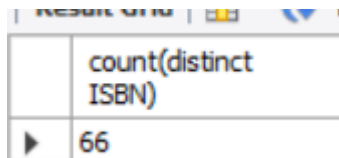
## PART 2: Data Manipulation

```
use Bookstore_Database;
```

```
-- total no of books
```

```
select count(distinct ISBN) from books_info;
```

Output:



	count(distinct ISBN)
►	66

Fig 1: Total number of recorded books in the database.

```
-- top 10 rated books
```

```
select title, author, rating from Books_info
```

```
group by title
```

```
order by rating desc limit 10;
```

Output:

title	author	rating
The Hobbit	J.R.R. Tolkien	9.40
To Kill a Mockingbird	Harper Lee	9.30
Atomic Habits	James Clear	9.30
The Adventures of Sherlock Holmes	Arthur Conan Doyle	9.20
War and Peace	Leo Tolstoy	9.20
Becoming	Michelle Obama	9.20
The Grapes of Wrath	John Steinbeck	9.10
Pride and Prejudice	Jane Austen	9.10
Sapiens	Yuval Noah Harari	9.10
The Body Keeps the Score	Bessel van der Kolk	9.10

Fig 2: top 10 rated books

-- 10 least rated books

select title, author, rating from Books\_info

group by title

order by rating asc limit 10;

Output:

	title	author	rating
▶	Train to Pakistan	Khushwant Singh	4.20
	Make Time	James Clear	4.30
	The Shadow Lines	Amitav Ghosh	4.30
	The Bachelor of Arts	R.K. Narayan	4.30
	The Palace of Illusions	Chitra Banerjee Divakaruni	4.40
	Inferno	Dan Brown	4.40
	The Namesake	Jhumpa Lahiri	4.50
	Brida	Paulo Coelho	4.50
	The Guide	R.K. Narayan	4.50
	The Lowland	Jhumpa Lahiri	4.50

Fig 3: 10 least rated books

-- oldest published book

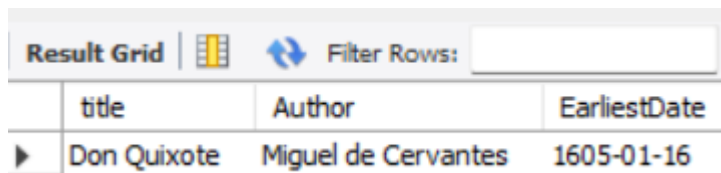
SELECT title, Author, MIN(pub\_Date) AS EarliestDate

FROM Bookstore\_Database.Books\_info

group by title

order by EarliestDate limit 1;

Output:



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains one row of data with three columns: 'title', 'Author', and 'EarliestDate'. The data row shows 'Don Quixote' by 'Miguel de Cervantes' published on '1605-01-16'.

	title	Author	EarliestDate
▶	Don Quixote	Miguel de Cervantes	1605-01-16

Fig 4: oldest published book

-- Latest published book

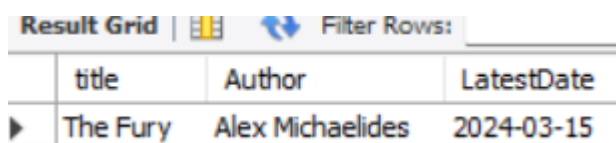
```
SELECT title, Author, min(pub_Date) AS LatestDate
```

```
FROM Bookstore_Database.Books_info
```

```
group by title
```

```
order by LatestDate desc limit 1;
```

Output:



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains one row of data with three columns: 'title', 'Author', and 'LatestDate'. The data row shows 'The Fury' by 'Alex Michaelides' published on '2024-03-15'.

	title	Author	LatestDate
▶	The Fury	Alex Michaelides	2024-03-15

Fig 5: Latest published book

-- Selecting the top 10 expensive books

```
select title, author, rating from Books_info
```

```
group by title
```

```
order by price desc limit 10;
```

Output:

title	author	rating
A Suitable Boy	Vikram Seth	4.70
Midnight's Children	Salman Rushdie	4.90
The God of Small Things	Arundhati Roy	4.80
The Silent Patient	Alex Michaelides	8.60
The Inheritance of Loss	Kiran Desai	4.70
The Guide	R.K. Narayan	4.50
Becoming	Michelle Obama	9.20
The Goldfinch	Donna Tartt	7.80
The Subtle Art of Not Giving a F*ck	Mark Manson	8.50
The White Tiger	Aravind Adiga	4.60

Fig 6: top 10 expensive books

-- selecting 10 least expensive books

select title, author, rating from Books\_info

group by title

order by price asc limit 10;

Output:

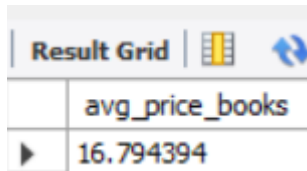
title	author	rating
Frankenstein	Mary Shelley	8.90
The Catcher in the Rye	J.D. Salinger	8.60
1984	George Orwell	9.00
To Kill a Mockingbird	Harper Lee	9.30
Journey to the Center of the Earth	Jules Verne	8.70
The Adventures of Sherlock Holmes	Arthur Conan Doyle	9.20
The Hobbit	J.R.R. Tolkien	9.40
The Picture of Dorian Gray	Oscar Wilde	8.60
Moby Dick	Herman Melville	8.10
Wuthering Heights	Emily Brontë	8.40

Fig 7: 10 least expensive books

-- Finding the average price of all books

```
select avg(Price) as avg_price_books  
from Books_info;
```

Output:



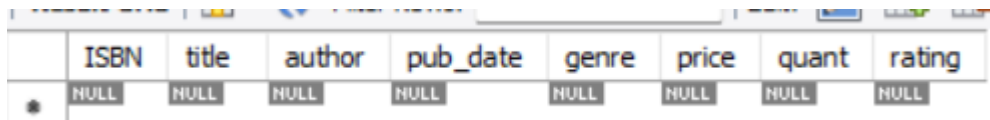
	avg_price_books
▶	16.794394

Fig 8: average price of all books

-- Finding duplicate values in books

```
select * from Books_info  
group by title  
having count(title) > 1;
```

Output:



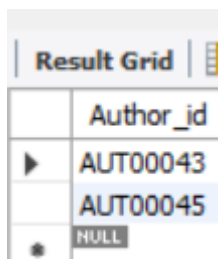
	ISBN	title	author	pub_date	genre	price	quant	rating
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 9: Finding duplicate values in books

-- Finding and removing duplicate from Authors table

```
select Author_id from Authors where Author = 'Vikram Seth';  
delete from Authors where Author_id = 'AUT00043';
```

Output:



	Author_id
▶	AUT00043
	AUT00045
*	NULL

Fig 10: Finding and removing duplicate from Authors table



```
-- Updating values in Orders table where null
select Order_id from Orders where Rev is null;

update Orders set Rev = Price * Quant where Order_id = 10002;
```

Output:

Order_id	cus_id	ISBN	Price	Quant	Rev
10001	104	ISBN00000002	9.99	5.00	49.95
10002	118	ISBN00000047	20.99	8.00	167.92

Fig 11: Updating values in Orders table where null

```
-- Finding the Titles starting with 'The'
select title from books_info
where title like 'The%';
```

Output:

title
The Adventures of Sherlock Holm
The Alchemist
The Art of Noticing
The Bachelor of Arts
The Body Keeps the Score
The Brothers Karamazov
The Catcher in the Rye

Fig 12: Finding the Titles starting with 'The'

```
-- Showing Customer table
select * from Customer_Info;
```

Output:

cus_id	cus_name	phone	email	address
101	John Doe	+1 123-456-789	john.doe@gmail.com	123 Main St, Anytown, USA
102	Jane Smith	+44 20 1234 5678	jane.smith@gmail.co.uk	456 Elm Ave, London, UK
103	Alice Johnson	+81 3 1234 5678	alice.johnson@gmail.jp	789 Sakura St, Tokyo, Japan
104	Robert Lee	+49 30 1234 5678	robert.lee@gmail.de	987 Linden Strasse, Berlin, Germany
105	Maria Rodriguez	+34 91 123 4567	maria.rodriguez@gmail.es	654 Calle Real, Madrid, Spain

Fig 13: Showing Customer table

-- Creating a separate column from address for country

SELECT

SUBSTRING\_INDEX(address, ',', -1) AS LastPart

FROM Customer\_Info;

Output:



LastPart
USA
UK
Japan
Germany
Spain
South Korea
China
Vietnam
Mexico
Australia
India
Vietnam
South Korea

Fig 14: Splitting country from address

-- Grouping customers data based in country of existence

SELECT

SUBSTRING\_INDEX(address, ',', -1) AS LastPart,

cus\_id,

cus\_name

FROM Customer\_Info

GROUP BY LastPart, cus\_id, cus\_name

order by LastPart;

Output:

LastPart	cus_id	cus_name
Australia	110	William Browne
Australia	125	Sophia Kim
China	107	Emily Chen
China	114	Liam Wang
China	122	Oliver Garcia
China	129	Isabella Patel
Germany	104	Robert Lee
Germany	119	Emma Davis
India	111	Olivia Patel
India	126	Lucas Lee
Japan	103	Alice Johnson
Japan	118	Liam Chen

Fig 15: Grouping customer names based on country

-- Finding the country that has the maximum numbers of customers

```

SELECT
    LastPart,
    COUNT(*) AS num_records
FROM (
    SELECT
        SUBSTRING_INDEX(address, ',', -1) AS LastPart
    FROM Customer_Info
) AS derived_table
GROUP BY LastPart
ORDER BY count(LastPart) desc;

```

Output:

	LastPart	num_records
▶	South Korea	4
	China	4
	Vietnam	4
	USA	2
	UK	2
	Japan	2
	Germany	2
	Spain	2
	Mexico	2
	Australia	2
	India	2
	Singapore	2

Fig 16: Finding the country that has the maximum numbers of customers

-- Rectifying wrong entry

update Authors set Author\_id = 'AUT00004'

where Author\_id = 'AUT000014';

select \* from Authors where Author\_id = 'AUT00004';

Output:

	Author_id	author	No_of_books	Rating	Country
▶	AUT00004	George Orwell	1	9.00	United Kingdom
•	NULL	NULL	NULL	NULL	NULL

Fig 17: Rectifying wrong entry

-- Finding top 5 Authors and last 5 Authors

SELECT Author, MAX(Rating) AS MaxRating

FROM Authors

GROUP BY Author

ORDER BY MaxRating DESC limit 5;

Output:

Result Grid			Filter Rows:
	Author	MaxRating	
▶	J.R.R. Tolkien	9.40	
	James Clear	9.30	
	Harper Lee	9.30	
	Arthur Conan Doyle	9.20	
	Leo Tolstoy	9.20	

Fig 18: Top 5 authors

```
SELECT Author, MAX(Rating) AS MaxRating
```

```
FROM Authors
```

```
GROUP BY Author
```

```
ORDER BY MaxRating ASC limit 5;
```

Output:

	Author	MaxRating
	Arundhati Roy	6.90
	Aravind Adiga	7.00
	Vikram Seth	7.10
	Jhumpa Lahiri	7.30
	Dan Brown	7.50

Fig 19: Last 5 authors

```
-- Finding Average rating of all Authors
```

```
SELECT avg(Rating) AS AvgRating
```

```
FROM Authors;
```

Output:

Result Grid	
	AvgRating
▶	8.421569

Fig 20: Average rating of all books

```
-- Showing Publishings table
```

```
select * from Publishers;
```

```
update Publishers set Pub_id = 'PUB066' where title = 'Malgudi Days';
```

Output:

Pub_id	Pub_name	title
PUB001	Secker & Warburg	Journey to the Center of the Earth
PUB002	J. B. Lippincott & Co.	Pride and Prejudice
PUB003	J. B. Lippincott & Co.	To Kill a Mockingbird
PUB004	Secker & Warburg	1984
PUB005	Secker & Warburg	The Great Gatsby
PUB006	Secker & Warburg	Moby Dick

Fig 21: Showing publishers table

-- Finding the publisher publishing the max number of books

```
select Pub_name, count(Pub_name) from Publishers  
group by Pub_name  
order by count(Pub_name) desc;
```

Output:

Pub_name	count(Pub_name)
Secker & Warburg	8
Random House	8
Penguin Books Ltd	6
HarperCollins	5
Celadon Books	3
J. B. Lippincott & Co.	2
Chatto & Windus	2
Vintage	2
Penguin Books	2
Viking Press	2
Doubleday Publishing	2
The Viking Press	1

Fig 22: Finding the publisher publishing the max number of books

-- Showing \* from Orders

```
select * from Orders;
```

Output:

Order_id	cus_id	ISBN	Price	Quant	Rev	total_bill_GST
10001	104	ISBN00000002	9.99	5.00	49.95	58.94
10002	118	ISBN00000047	20.99	8.00	167.92	198.15
10003	101	ISBN00000002	12.99	6.00	77.94	91.97
10004	102	ISBN00000034	20.99	3.00	62.97	74.30
10005	103	ISBN00000008	6.99	2.00	13.98	16.50
10006	103	ISBN00000015	16.99	1.00	16.99	20.05
10007	102	ISBN00000041	19.99	2.00	39.98	47.18
10008	104	ISBN00000041	19.99	1.00	19.99	23.59
10009	106	ISBN00000042	24.99	1.00	24.99	29.49
10010	106	ISBN00000017	12.99	12.00	155.88	183.94

Fig 23: Orders data

-- Book generating the maximum revenues

Select \* from Orders

group by Order\_id

order by Rev desc;

Output:

Order_id	cus_id	ISBN	Price	Quant	Rev
10015	110	ISBN00000044	21.99	11.00	241.89
10016	111	ISBN00000019	11.99	20.00	239.80
10002	118	ISBN00000047	20.99	8.00	167.92
10030	120	ISBN00000060	8.00	19.99	159.92
10010	106	ISBN00000017	12.99	12.00	155.88
10012	108	ISBN00000019	11.99	10.00	119.90

Fig 24: Book generating the maximum revenues

-- List of 10 people with most spending

Select cus\_id, max(Rev) from Orders

group by Order\_id

order by Rev desc limit 10;

Output:

	cus_id	max(Rev)
▶	110	241.89
	111	239.80
	118	167.92
	120	159.92
	106	155.88
	108	119.90

Fig 25: -- List of 10 people with most spending

-- Calculating the total revenue

select sum(Rev) from Orders;

Output:

	sum(Rev)
▶	2080.74

Fig 26: Calculating the total revenue

-- cross join

select \* from

Customer\_Info

cross join Orders;

Output:

	cus_id	cus_name	phone	email	address	Order_id	cus_id	ISBN	Price	Quant	Rev
▶	130	Alexander Tan	+65 6 9876 5432	alexander.tan@gmail.sg	789 Orchard Rd, Singapore	10001	104	ISBN00000002	9.99	5.00	49.95
	129	Isabella Patel	+86 21 9876 5432	isabella.patel@gmail.cn	456 Nanjing Rd, Hangzhou, China	10001	104	ISBN00000002	9.99	5.00	49.95
	128	Mason Nguyen	+82 51 9876 5432	mason.nguyen@gmail.kr	321 Haeundae Beach Rd, Busan, South Korea	10001	104	ISBN00000002	9.99	5.00	49.95
	127	Olivia Martinez	+84 28 9876 5432	olivia.martinez@gmail.vn	987 Le Loi St, Da Nang, Vietnam	10001	104	ISBN00000002	9.99	5.00	49.95
	126	Lucas Lee	+91 22 9876 5432	lucas.lee@gmail.in	654 MG Road, Bangalore, India	10001	104	ISBN00000002	9.99	5.00	49.95
	125	Sophia Kim	+61 2 9876 5432	sophia.kim@example.au	123 George St, Melbourne, Australia	10001	104	ISBN00000002	9.99	5.00	49.95

Fig 27: Cross join of Customers data and Orders data

It is not very useful in real-life scenarios and makes no proper sense except during matrix calculation

-- Left Join Joining orders data and customers data

select Customer\_Info.cus\_name, Customer\_Info.address, Customer\_Info.phone,



Orders.Order\_id, Orders.ISBN, Orders.Rev

from Customer\_Info

left join Orders on Orders.cus\_id= Customer\_Info.cus\_id;

Output:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:		
	cus_name	address	phone	Order_id	ISBN	Rev
▶	John Doe	123 Main St, Anytown, USA	+1 123-456-789	10003	ISBN00000002	77.94
	Jane Smith	456 Elm Ave, London, UK	+44 20 1234 5678	10004	ISBN00000034	62.97
	Jane Smith	456 Elm Ave, London, UK	+44 20 1234 5678	10007	ISBN00000041	39.98
	Alice Johnson	789 Sakura St, Tokyo, Japan	+81 3 1234 5678	10005	ISBN00000008	13.98
	Alice Johnson	789 Sakura St, Tokyo, Japan	+81 3 1234 5678	10006	ISBN00000015	16.99
	Robert Lee	987 Linden Strasse, Berlin, Germany	+49 30 1234 5678	10001	ISBN00000002	49.95
	Robert Lee	987 Linden Strasse, Berlin, Germany	+49 30 1234 5678	10008	ISBN00000041	19.99

Fig 28: Left join of customers and Orders data.

-- Right join

select Customer\_Info.cus\_name, Customer\_Info.address, Customer\_Info.phone,

Orders.Order\_id, Orders.ISBN, Orders.Rev

from Customer\_Info

right join Orders on Orders.cus\_id= Customer\_Info.cus\_id;

Output:

	cus_name	address	phone	Order_id	ISBN	Rev
▶	Robert Lee	987 Linden Strasse, Berlin, Germany	+49 30 1234 5678	10001	ISBN00000002	49.95
	Liam Chen	123 Sakura St, Kyoto, Japan	+81 3 9876 5432	10002	ISBN00000047	167.92
	John Doe	123 Main St, Anytown, USA	+1 123-456-789	10003	ISBN00000002	77.94
	Jane Smith	456 Elm Ave, London, UK	+44 20 1234 5678	10004	ISBN00000034	62.97
	Alice Johnson	789 Sakura St, Tokyo, Japan	+81 3 1234 5678	10005	ISBN00000008	13.98
	Alice Johnson	789 Sakura St, Tokyo, Japan	+81 3 1234 5678	10006	ISBN00000015	16.99

Fig 29: Right join of customers and Orders data.

-- Ordering Orders data by Revenue

Select cus\_id from Orders

group by Order\_id

order by Rev desc limit 0;

select \* from Orders

order by rev desc;

Output:

	Order_id	cus_id	ISBN	Price	Quant	Rev
►	10015	110	ISBN0000044	21.99	11.00	241.89
	10016	111	ISBN0000019	11.99	20.00	239.80
	10002	118	ISBN0000047	20.99	8.00	167.92
	10030	120	ISBN0000060	8.00	19.99	159.92
	10010	106	ISBN0000017	12.99	12.00	155.88
	10012	108	ISBN0000019	11.99	10.00	119.90
	10020	115	ISBN0000050	6.00	17.00	102.00

Fig 30: Ordering Orders data by the revenue

-- Creating a case for discounts

SELECT

Rev,

CASE

WHEN Rev BETWEEN 20 AND 25 THEN Rev - (0.02 \* Rev)

WHEN Rev BETWEEN 25 AND 30 THEN Rev - (0.03 \* Rev)

WHEN Rev BETWEEN 30 AND 50 THEN Rev - (0.05 \* Rev)

WHEN Rev BETWEEN 50 AND 100 THEN Rev - (0.1 \* Rev)

WHEN Rev > 100 THEN Rev - (0.15 \* Rev)

ELSE 'No Discount Applicable'

END AS DiscountedRev

FROM Orders;

Output:

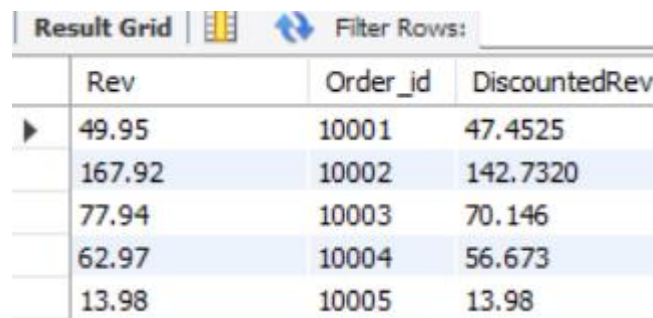
	Rev	DiscountedRev
►	49.95	47.4525
	167.92	142.7320
	77.94	70.146
	62.97	56.673
	13.98	No Discount Applicable
	16.99	No Discount Applicable
	39.98	37.9810
	19.99	No Discount Applicable
	24.99	24.4902
	155.88	132.4980

Fig 31: Discounting Procedure

-- returning the original amount

```
SELECT
    Rev, Order_id,
    CASE
        WHEN Rev BETWEEN 20 AND 25 THEN Rev - (0.02 * Rev)
        WHEN Rev BETWEEN 25 AND 30 THEN Rev - (0.03 * Rev)
        WHEN Rev BETWEEN 30 AND 50 THEN Rev - (0.05 * Rev)
        WHEN Rev BETWEEN 50 AND 100 THEN Rev - (0.1 * Rev)
        WHEN Rev > 100 THEN Rev - (0.15 * Rev)
        ELSE Rev
    END AS DiscountedRev
FROM Orders;
```

Output:



	Rev	Order_id	DiscountedRev
▶	49.95	10001	47.4525
	167.92	10002	142.7320
	77.94	10003	70.146
	62.97	10004	56.673
	13.98	10005	13.98

Fig 32: -- returning the original amount

-- Create procedure taxes

```
create table added_tax
(
    Order_id int,
    DiscountedRev decimal(10, 2),
    tax_rate decimal(4,1),
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
DELIMITER //
```

```

CREATE PROCEDURE taxes
( IN p_Order_id int, IN p_DiscountedRev DECIMAL(10, 2), IN p_tax_rate decimal(4,1) )
BEGIN
    DECLARE p_tax DECIMAL(10, 2);

    SET p_tax = p_DiscountedRev * (p_tax_rate / 100);

    SELECT
        p_DiscountedRev AS OriginalAmount,
        p_tax AS taxAmount,
        p_DiscountedRev + p_tax AS TotalAmount;
END //

DELIMITER ;
CALL taxes(10003, 70.146, 18.2);

```

Output:

	OriginalAmount	taxAmount	TotalAmount
▶	70.15	12.77	82.92

Fig 33: Taxes Output

-- creating a trigger

```
select * from Orders;
```

```
select * from customer_info;
```

```
alter table Customer_Info add wallet_vale decimal(10,2);
```

```
update Customer_Info set wallet_vale = 100.00 where wallet_vale is null;
```

```
CREATE TABLE transactions (
```

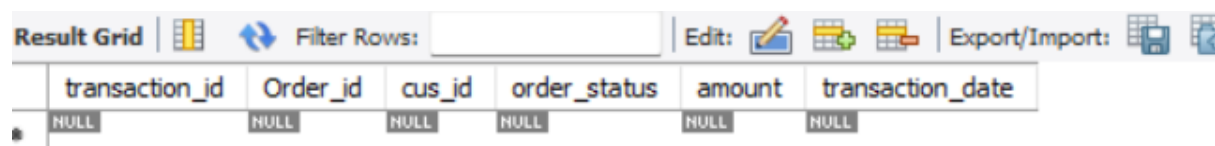
```

transaction_id INT AUTO_INCREMENT PRIMARY KEY,
Order_id int,
cus_id INT NOT NULL,
order_status ENUM('Delivered', 'Not-Delivered') NOT NULL,
amount DECIMAL(15, 2) NOT NULL,
transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (cus_id) REFERENCES Customer_Info(cus_id),
FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);

```

```
select * from transactions;
```

Output:



transaction_id	Order_id	cus_id	order_status	amount	transaction_date
NULL	NULL	NULL	NULL	NULL	NULL

Fig 34: Transaction dable

```
DELIMITER //
```

```
CREATE PROCEDURE insert_transaction
```

```

( IN p_Order_id INT, p_cus_id int, IN p_order_status ENUM('Delivered', 'Not-Delivered'),
IN p_amount DECIMAL(15, 2) )

```

```
begin
```

```
    DECLARE current_wallet_vale DECIMAL(15, 2);
```

```
    SELECT wallet_vale INTO current_wallet_vale FROM Customer_Info WHERE cus_id =
p_cus_id;
```

```
    INSERT INTO transactions (Order_id, cus_id, order_status, amount) VALUES
(p_Order_id, p_cus_id, p_order_status, p_amount);
```

```
    if p_order_status = 'Delivered' then
```

```
        update Customer_Info set wallet_vale = wallet_vale - p_amount where
cus_id=p_cus_id;
```

```
    elseif p_transaction_type = 'Not-Delivered' then
```

```

        if (current_wallet_vale-p_amount) < 0 then
            select 'Insufficient balance for purchase';
        else
            update Customer_Info set wallet_vale=wallet_vale-p_amount where
cus_id=p_cus_id;
        end if;
    end if;
end //

```

DELIMITER ;

```
CALL insert_transaction(10004, 102,'Delivered', 74.30);
```

```

-- creating trigger assuming that the amount is debited at the time of ordering and the
-- balance status changes depending on the order status being delivered or not delivered.
alter table transactions add cancel_flag varchar(5);
update transactions set cancel_flag='N';

```

```
delimiter // -- two types of triggers after and before insert
```

```
    -- (trigger name)
```

```
create trigger check_flag after update on transactions for each row
```

```
begin
```

```
if new.cancel_flag='Y' and old.cancel_flag != new.cancel_flag then
```

```
    if old.order_status = 'Delivered' then
```

```
        update Customer_Info set wallet_vale = wallet_vale + old.amount where cus_id =
old.cus_id;
```

```
    else if old.tranction_type='Not-Delivered' then
```

```
        update Customer_Info set wallet_vale = wallet_vale + old.amount where cus_id =
old.cus_id;
```

```

end if;

else if new.cancel_flag = 'N' then

if old.transaaction_type = 'Delivered' then

    update Customer_Info set wallet_vale = old.wallet_vale where cus_id = old.cus_id;

    else if old.tranction_type='Not-Delivered' then

        update Customer_Info set wallet_vale = wallet_vale + old.amount where cus_id =
old.cus_id;

        end if;

end if ;

end //

```

delimiter ;

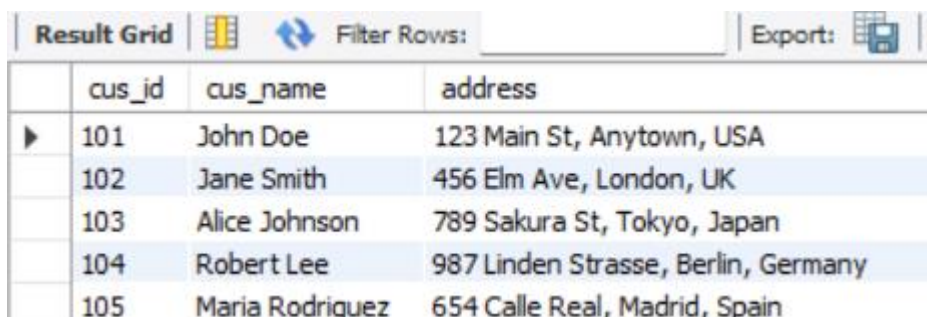
-- creating views

create view basic\_Customer\_Info\_view as

select cus\_id, cus\_name, address from Customer\_Info;

select \* from basic\_Customer\_Info\_view;

Output:

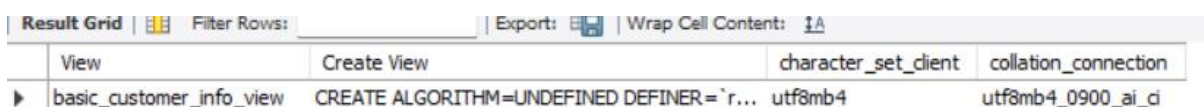


	cus_id	cus_name	address
▶	101	John Doe	123 Main St, Anytown, USA
	102	Jane Smith	456 Elm Ave, London, UK
	103	Alice Johnson	789 Sakura St, Tokyo, Japan
	104	Robert Lee	987 Linden Strasse, Berlin, Germany
	105	Maria Rodriguez	654 Calle Real, Madrid, Spain

Fig 35: View format of Coustomers data

show create view basic\_Customer\_Info\_view;

Output:



View	Create View	character_set_client	collation_connection
▶ basic_customer_info_view	CREATE ALGORITHM=UNDEFINED DEFINER=`r...`	utf8mb4	utf8mb4_0900_ai_ci

Fig 36: Show create view

-- Grant and Revoke Comnds

grant select, insert, delete, update on Authors to 'John'@'Localhost';

Output: 12:29:07      grant select, insert, delete, update on Authors to 'John'@'Localhost' 0  
row(s) affected      0.031 sec

Revoke delete, update on Authors from 'John'@'Localhost';

Output: 12:29:48      Revoke delete, update on Authors from 'John'@'Localhost' 0 row(s)  
affected      0.000 sec

Commit;

SET SQL\_SAFE\_UPDATES = 0;

drop database Bookstore\_Database;

Thank You