Online Bookstore Database Management System

Project Scope:

Design and implement a SQL database for an online bookstore. The database should handle various aspects of book inventory, customer orders, and administrative functions.

Steps to Implement the Project:

1. Database Design (DDL - Data Definition Language):
   * Step 1: Identify and define the entities (tables) needed for the bookstore database. Examples include:
     + Books (ISBN, title, author, publisher, genre, price, etc.)
     + Customers (customer\_id, name, address, email, etc.)
     + Orders (order\_id, customer\_id, order\_date, status, total\_amount, etc.)
     + Order Details (order\_id, ISBN, quantity, price)
     + Employees (employee\_id, name, position, etc.)
     + Categories, Publishers, Reviews, etc.
   * Step 2: Define the relationships between the tables (primary keys, foreign keys).
   * Step 3: Implement the schema using SQL commands (`CREATE TABLE`, `ALTER TABLE` for constraints, etc.).
2. Populating Data (DML - Data Manipulation Language):
   * Step 4: Insert sample data into the tables to populate them (`INSERT INTO` statements).
3. Querying Data (DQL - Data Query Language):
   * Step 5: Write SQL queries to retrieve information such as:
     + List of books by a specific author.
     + Total sales by month.
     + Customers who purchased more than a certain amount.
     + Best-selling books.
     + Pending orders.
     + etc.
4. Advanced Queries and Subqueries:
   * Step 6: Incorporate subqueries to solve complex queries. For example:
     + Find customers who have placed orders exceeding a certain total amount.
     + Identify books that have never been ordered.
5. Using Joins:
   * Step 7: Utilize different types of joins (`INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`) to combine data from multiple tables. For example:
     + Retrieve order details along with customer information.
     + List books along with their publishers.
6. Creating Views:
   * Step 8: Design and create views for commonly used queries. For instance:
     + Create a view to show all orders along with customer details.
     + Create a view for books that are currently in stock.
7. Testing and Optimization:
   * Step 9: Test the database queries and ensure they return accurate results.
   * Step 10: Optimize queries for better performance if necessary (indexing, query restructuring).
8. Documentation:
   * Step 11: Document the database schema, sample queries, and any specific business rules or assumptions made during the design phase by making ER Diagram.

QUERIES:

1. CREATING DATABASE:

CREATE DATABASE BOOKSTORE; USE BOOKSTORE;

1. CREATING TABLES:

CREATE TABLE Publishers (

publisher\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255) NOT NULL,

address VARCHAR(255), contact\_number VARCHAR(15), email VARCHAR(255) UNIQUE

);

CREATE TABLE Books (

ISBN VARCHAR(13) PRIMARY KEY, title VARCHAR(255) NOT NULL, author VARCHAR(255) NOT NULL,

publisher\_id INT, genre VARCHAR(255),

price DECIMAL(10, 2) NOT NULL,

publication\_year INT, stock\_quantity INT,

FOREIGN KEY (publisher\_id) REFERENCES Publishers(publisher\_id)

);

CREATE TABLE Customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255) NOT NULL,

address VARCHAR(255),

email VARCHAR(255) UNIQUE,

phone\_number VARCHAR(15), registration\_date DATE

);

CREATE TABLE Orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT, order\_date DATE NOT NULL, status VARCHAR(50),

total\_amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

CREATE TABLE Order\_Details ( order\_id INT,

ISBN VARCHAR(13),

quantity INT NOT NULL,

price DECIMAL(10, 2) NOT NULL,

PRIMARY KEY (order\_id, ISBN),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id), FOREIGN KEY (ISBN) REFERENCES Books(ISBN)

);

CREATE TABLE Employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255) NOT NULL,

position VARCHAR(255), hire\_date DATE,

salary DECIMAL(10, 2)

);

CREATE TABLE Categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY, category\_name VARCHAR(255) NOT NULL, description TEXT

);

CREATE TABLE Reviews (

review\_id INT AUTO\_INCREMENT PRIMARY KEY, ISBN VARCHAR(13),

customer\_id INT,

rating INT CHECK (rating >= 1 AND rating <= 5), review\_text TEXT,

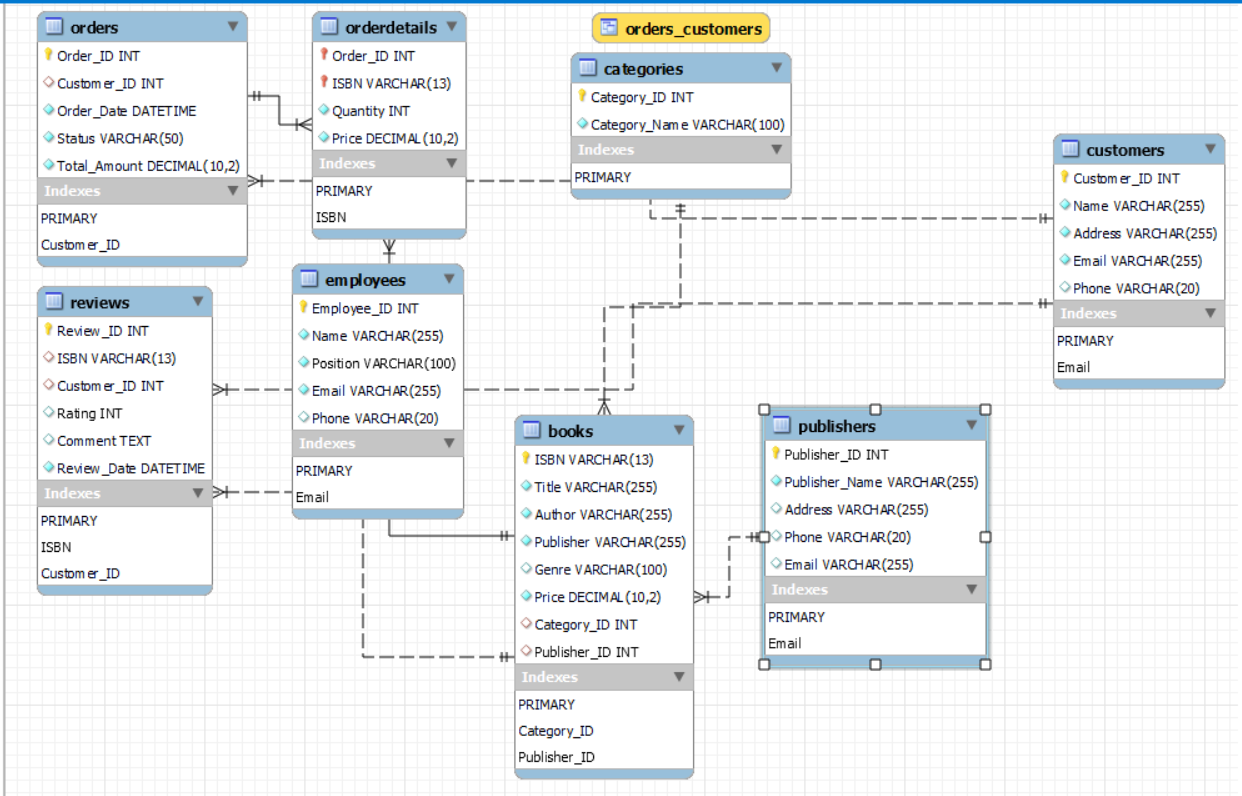
review\_date DATE,

FOREIGN KEY (ISBN) REFERENCES Books(ISBN),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

1. ER DIAGRAM:



1. INSERTINTG VALUES IN TABLES:

INSERT INTO Publishers (Publisher\_Name, Address, Phone, Email) VALUES

('Penguin Random House', '1745 Broadway, New York, NY 10019', '212-782-9000', 'info@penguinrandomhouse.com'),

('HarperCollins', '195 Broadway, New York, NY 10007', '212-207-7000', 'info@harpercollins.com'),

('Simon & Schuster', '1230 Avenue of the Americas, New York, NY 10020', '212-698-7000', 'info@simonandschuster.com'),

('Hachette Book Group', '1290 Avenue of the Americas, New York, NY 10104', '212-364-1100', 'info@hachettebookgroup.com'),

('Macmillan Publishers', '120 Broadway, New York, NY 10271', '646-307-5151', 'info@macmillan.com'),

('Scholastic', '557 Broadway, New York, NY 10012', '212-343-6100', 'info@scholastic.com'),

('Cengage', '200 Pier 4 Blvd, Boston, MA 02210', '800-354-9706', 'info@cengage.com'),

('Pearson', '221 River St, Hoboken, NJ 07030', '201-236-7000', 'info@pearson.com'),

('Wiley', '111 River St, Hoboken, NJ 07030', '201-748-6000', 'info@wiley.com'),

('Reilly Media', '1005 Gravenstein Hwy North, Sebastopol, CA 95472', '707-827-7000', 'info@oreilly.com');

INSERT INTO Books (ISBN, Title, Author, Publisher, Genre, Price, Category\_ID, Publisher\_ID) VALUES

('9780143128540', 'The Great Gatsby', 'F. Scott Fitzgerald', 'Penguin Random House', 'Fiction', 10.99, 1, 1),

('9780062316097', 'To Kill a Mockingbird', 'Harper Lee', 'HarperCollins', 'Fiction', 7.99, 1, 2),

('9781501100335', 'The Martian', 'Andy Weir', 'Simon & Schuster', 'Science Fiction', 9.99, 3, 3),

('9780316417292', 'The Hobbit', 'J.R.R. Tolkien', 'Hachette Book Group', 'Fantasy', 8.99, 4, 4),

('9780307277671', 'The Da Vinci Code', 'Dan Brown', 'Penguin Random House', 'Mystery', 12.99, 5, 1),

('9781451648539', 'Steve Jobs', 'Walter Isaacson', 'Simon & Schuster', 'Biography', 15.99, 6, 3),

('9780307387943', 'Sapiens: A Brief History of Humankind', 'Yuval Noah Harari', 'HarperCollins', 'History', 14.99, 7, 2),

('9780545582889', 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 'Scholastic', 'Children', 6.99, 8, 6),

('9780345803481', 'Fifty Shades of Grey', 'E.L. James', 'Penguin Random House', 'Romance', 9.99, 9, 1),

('9780399184413', 'Atomic Habits', 'James Clear', 'Penguin Random House', 'Self-Help', 11.99, 10, 1);

INSERT INTO Customers (Name, Address, Email, Phone) VALUES

('John Doe', '123 Maple St, Springfield, IL 62704', 'johndoe@example.com', '217-555-1234'),

('Jane Smith', '456 Oak St, Springfield, IL 62704', 'janesmith@example.com', '217-555-5678'),

('Alice Johnson', '789 Pine St, Springfield, IL 62704', 'alicej@example.com', '217-555-9101'),

('Bob Brown', '101 Elm St, Springfield, IL 62704', 'bobbrown@example.com', '217-555-1122'),

('Charlie Davis', '202 Cedar St, Springfield, IL 62704', 'charlied@example.com', '217-555-3344'),

('Diana Evans', '303 Birch St, Springfield, IL 62704', 'dianae@example.com', '217-555-5566'),

('Ethan Harris', '404 Walnut St, Springfield, IL 62704', 'ethanh@example.com', '217-555-7788'),

('Fiona Green', '505 Chestnut St, Springfield, IL 62704', 'fionag@example.com', '217-555-9900'),

('George Hill', '606 Maple St, Springfield, IL 62704', 'georgeh@example.com', '217-555-2233'),

('Hannah Clark', '707 Oak St, Springfield, IL 62704', 'hannahc@example.com', '217-555-4455');

INSERT INTO Orders (Customer\_ID, Order\_Date, Status, Total\_Amount) VALUES

(1, '2023-01-15 10:00:00', 'Shipped', 29.97),

(2, '2023-02-20 11:30:00', 'Processing', 15.99),

(3, '2023-03-25 12:45:00', 'Delivered', 23.98),

(4, '2023-04-30 14:00:00', 'Canceled', 0.00),

(5, '2023-05-05 15:15:00', 'Shipped', 44.97),

(6, '2023-06-10 16:30:00', 'Delivered', 18.98),

(7, '2023-07-15 17:45:00', 'Processing', 14.99),

(8, '2023-08-20 19:00:00', 'Delivered', 19.98),

(9, '2023-09-25 20:15:00', 'Shipped', 30.97),

(10, '2023-10-30 21:30:00', 'Processing', 9.99);

INSERT INTO OrderDetails (Order\_ID, ISBN, Quantity, Price) VALUES

(1, '9780143128540', 1, 10.99),

(1, '9780062316097', 2, 7.99),

(2, '9781501100335', 1, 15.99),

(3, '9780316417292', 2, 8.99),

(3, '9780307277671', 1, 12.99),

(5, '9781451648539', 3, 15.99),

(6, '9780307387943', 2, 14.99),

(7, '9780545582889', 1, 6.99),

(8, '9780345803481', 2, 9.99),

(9, '9780399184413', 3, 11.99);

INSERT INTO Employees (Name, Position, Email, Phone) VALUES

('Alice Walker', 'Manager', 'alicew@example.com', '217-555-1212'),

('Bob Stone', 'Cashier', 'bobs@example.com', '217-555-2323'),

('Carol Baker', 'Stock Clerk', 'carolb@example.com', '217-555-3434'),

('David Lee', 'Customer Service', 'davidl@example.com', '217-555-4545'),

('Emma White', 'Cashier', 'emmaw@example.com', '217-555-5656'),

('Frank Green', 'Manager', 'frankg@example.com', '217-555-6767'),

('Grace Harris', 'Stock Clerk', 'graceh@example.com', '217-555-7878'),

('Henry Adams', 'Customer Service', 'henrya@example.com', '217-555-8989'),

('Isabel Young', 'Manager', 'isabely@example.com', '217-555-9090'),

('Jack Turner', 'Cashier', 'jackt@example.com', '217-555-1010');

INSERT INTO Categories (Category\_Name) VALUES

('Fiction'),

('Non-Fiction'),

('Science Fiction'),

('Fantasy'),

('Mystery'),

('Biography'),

('History'),

('Children'),

('Romance'),

('Self-Help');

INSERT INTO Reviews (ISBN, Customer\_ID, Rating, Comment, Review\_Date) VALUES

('9780143128540', 1, 5, 'An amazing book!', '2023-01-20 10:30:00'),

('9780062316097', 2, 4, 'Really enjoyed it.', '2023-02-25 11:45:00'),

('9781501100335', 3, 5, 'A fantastic read!', '2023-03-30 12:50:00'),

('9780316417292', 4, 3, 'Good, but a bit slow.', '2023-04-05 14:05:00'),

('9780307277671', 5, 4, 'Very engaging.', '2023-05-10 15:20:00'),

('9781451648539', 6, 5, 'Incredibly inspiring.', '2023-06-15 16:35:00'),

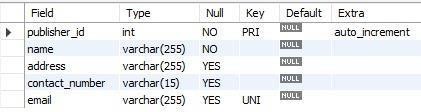
('9780307387943', 7, 4, 'Thought-provoking.', '2023-07-20 17:50:00'),

('9780545582889', 8, 5, 'Magical!', '2023-08-25 19:05:00'),

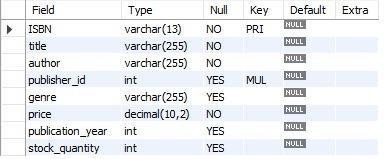
('9780345803481', 9, 3, 'Not my cup of tea.', '2023-09-30 20:20:00'),

('9780399184413', 10, 5, 'Life-changing.', '2023-10-05 21:35:00');

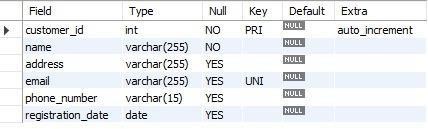
1. TABLE STRUCTURES:

DESC Publishers;

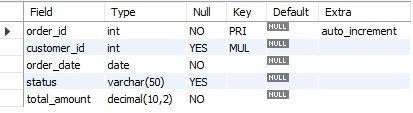
DESC Books;



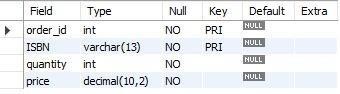
DESC Customers;



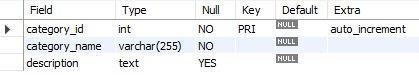
DESC Orders;



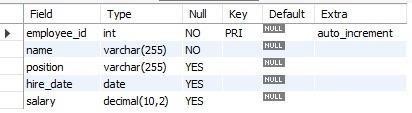
DESC Order\_Details;



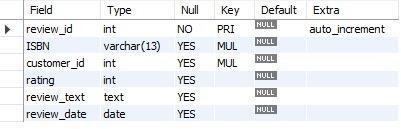
DESC Categories;



DESC Employees;

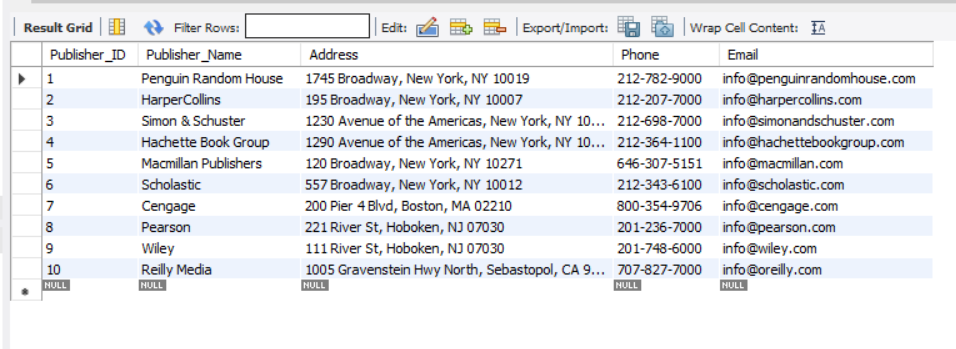


DESC Reviews;

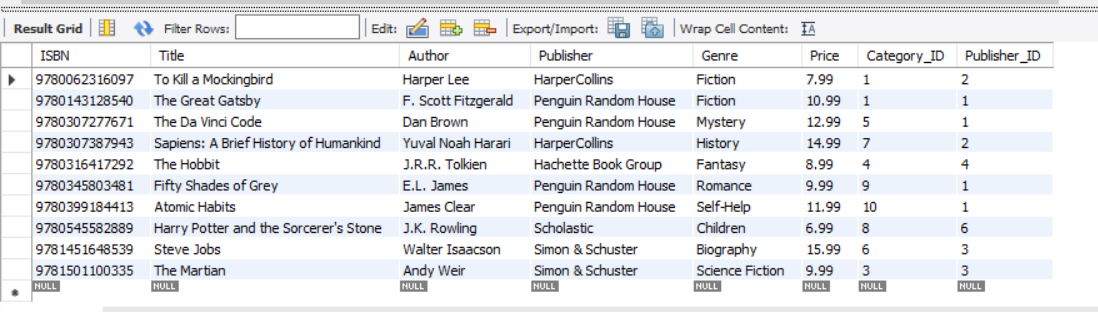


1. TABLE CONTENT:

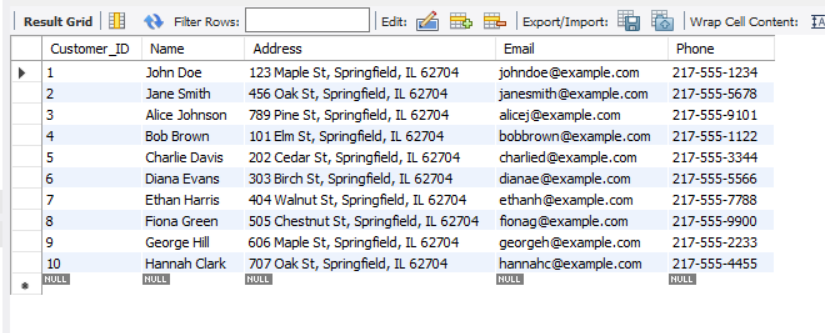
SELECT \* FROM Publishers;



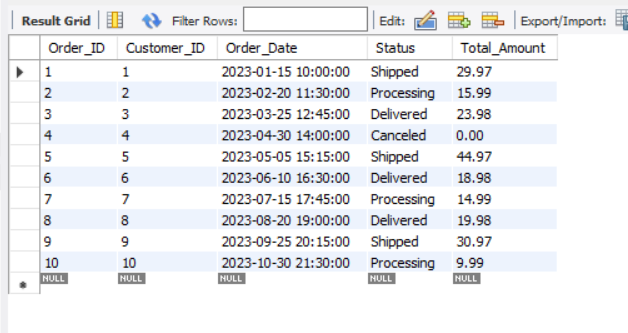
SELECT \* FROM Books;



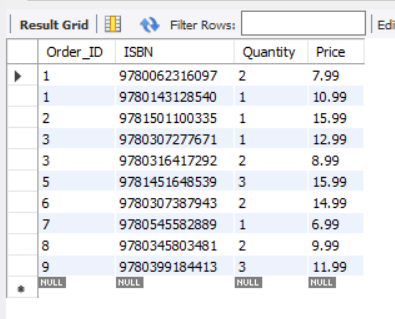
SELECT \* FROM Customers;



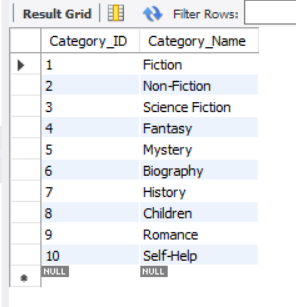
SELECT \* FROM Orders;



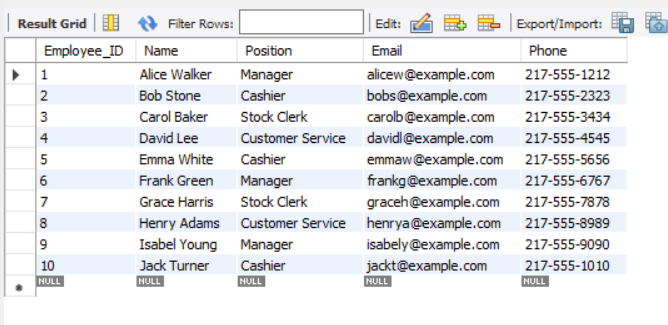
SELECT \* FROM Order\_Details;



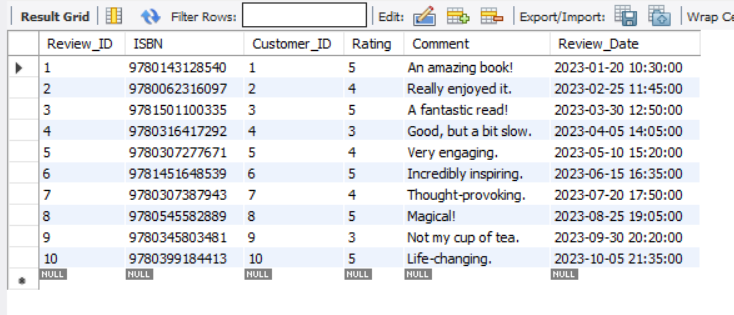
SELECT \* FROM Categories;



SELECT \* FROM Employees;

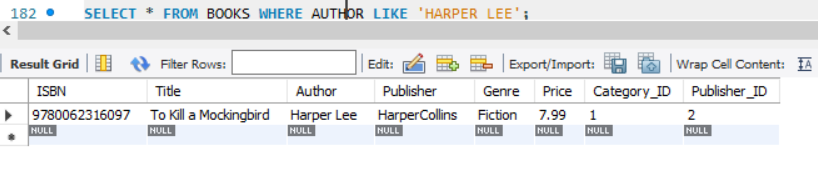


SELECT \* FROM Reviews;



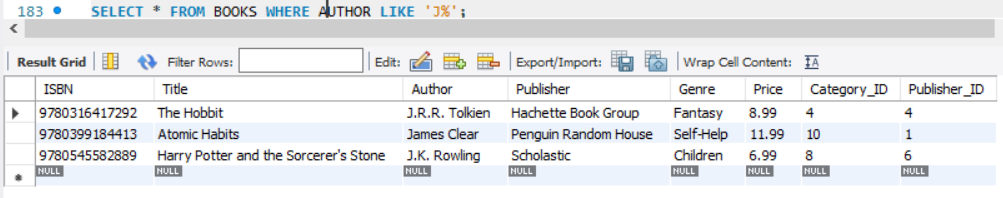
1. DQL QUERIES:
   1. List of Books by specific Author:

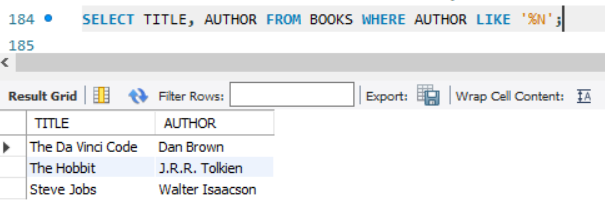
SELECT \* FROM BOOKS WHERE AUTHOR LIKE 'HARPER LEE';



* 1. List of Books where Author Name starts with ‘B’:

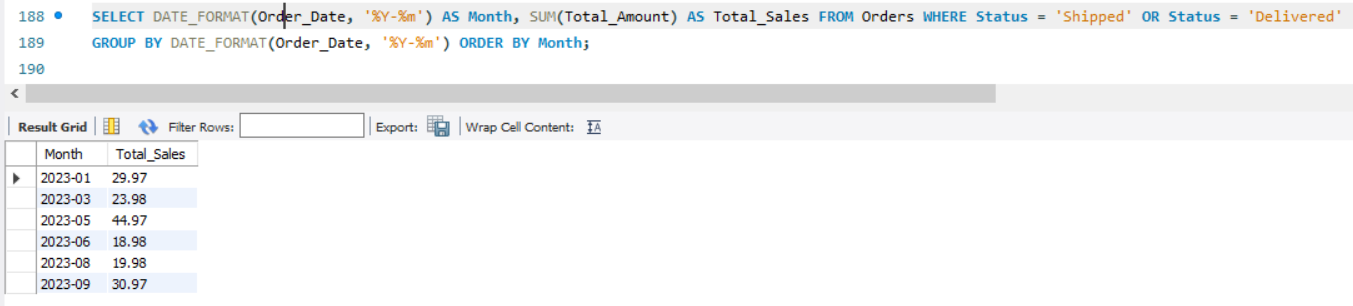
SELECT TITLE, AUTHOR, GENRE, PUBLICATION\_YEAR FROM BOOKS WHERE AUTHOR LIKE 'B%';





* 1. Total Sales by Month:

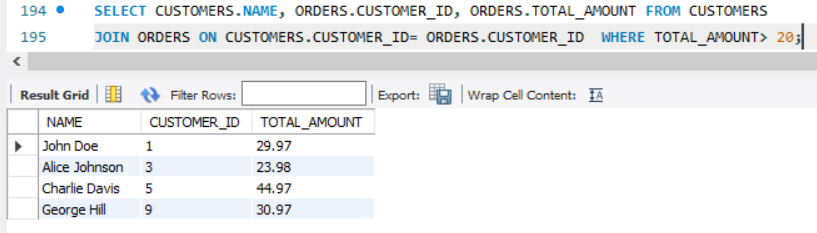
SELECT DATE\_FORMAT(Order\_Date, '%Y-%m') AS Month, SUM(Total\_Amount) AS Total\_Sales FROM Orders WHERE Status = 'Shipped' OR Status = 'Delivered' GROUP BY DATE\_FORMAT(Order\_Date, '%Y-%m') ORDER BY Month;



* 1. Customers who purchased more than a certain amount:

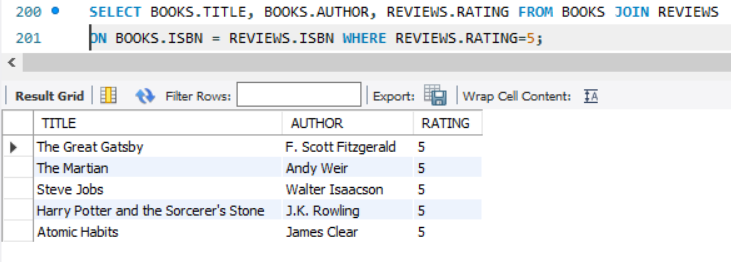
SELECT CUSTOMERS.NAME, ORDERS.CUSTOMER\_ID, ORDERS.TOTAL\_AMOUNT FROM CUSTOMERS JOIN ORDERS ON CUSTOMERS.CUSTOMER\_ID= ORDERS.CUSTOMER\_ID

WHERE TOTAL\_AMOUNT> 15;



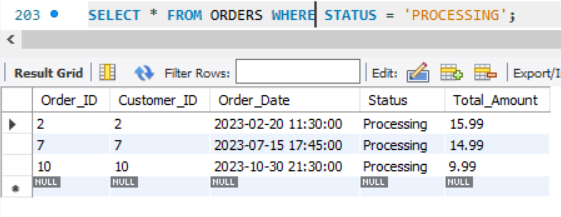
* 1. Best-Selling Books:

SELECT BOOKS.TITLE, BOOKS.AUTHOR, REVIEWS.RATING FROM BOOKS JOIN REVIEWS ON BOOKS.ISBN = REVIEWS.ISBN WHERE REVIEWS.RATING=5;



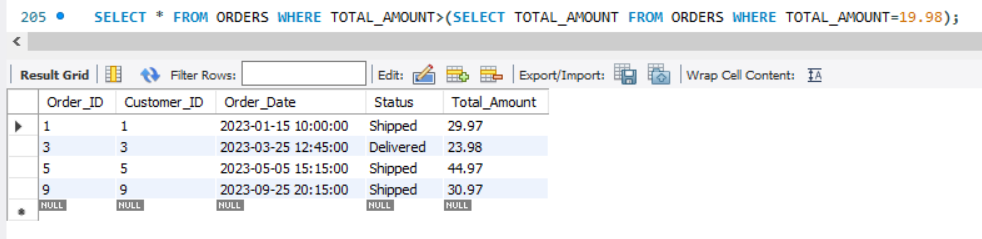
* 1. Pending Orders:

SELECT \* FROM ORDERS WHERE STATUS = 'PROCESSING';



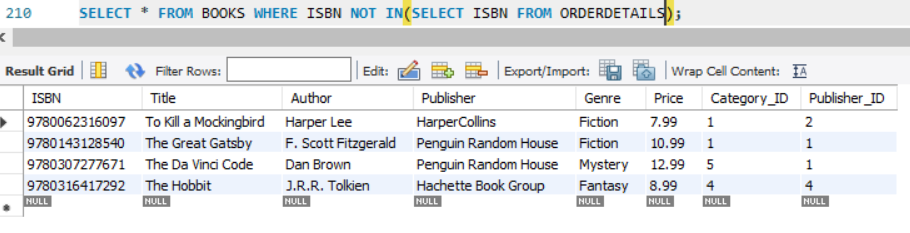
1. SUBQUERIES:
   1. Customers who have placed orders exceeding a amount of another customer:

SELECT \* FROM ORDERS WHERE TOTAL\_AMOUNT>(SELECT TOTAL\_AMOUNT FROM ORDERS WHERE TOTAL\_AMOUNT=19.98);

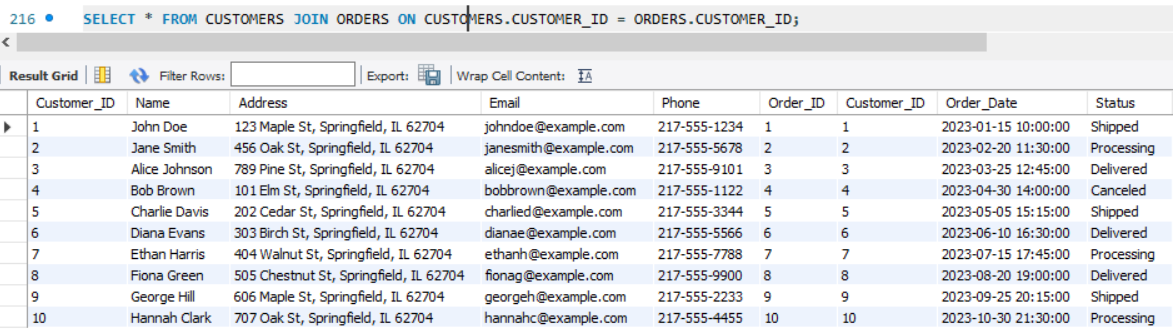


* 1. Books that haven’t been ordered:

SELECT \* FROM BOOKS WHERE ISBN NOT IN(SELECT ISBN FROM ORDER\_DETAILS);

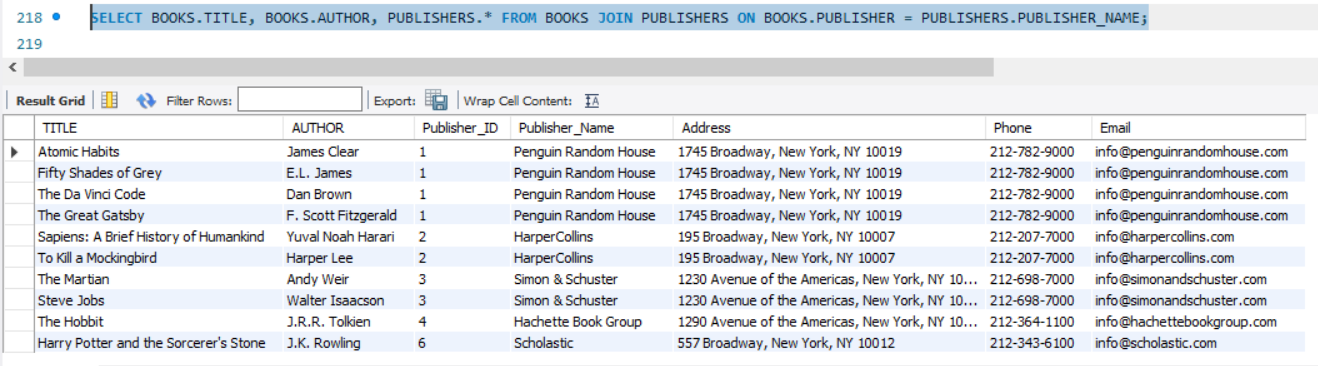


1. JOINS:
   1. Order Details with Customer Information:

SELECT \* FROM CUSTOMERS JOIN ORDERS ON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_ID; 

* 1. Book Details along with Publisher Details:

SELECT BOOKS.TITLE, BOOKS.AUTHOR, PUBLISHERS.\* FROM BOOKS JOIN PUBLISHERS ON BOOKS.PUBLISHER = PUBLISHERS.PUBLISHER\_NAME;



1. VIEWS:
   1. Creating a View containing Customer Order Details:

CREATE VIEW INVENTORY AS (SELECT BOOKS.TITLE AS TITLE, BOOKS.AUTHOR AS AUTHOR, BOOKS.GENRE AS GENRE, BOOKS.PUBLICATION\_YEAR AS PUBLICATION\_YEAR, BOOKS.PRICE AS PRICE, REVIEWS.RATING AS RATING, BOOKS.STOCK\_QUANTITY AS STOCK\_QUANTITY FROM BOOKS JOIN REVIEWS ON BOOKS.ISBN = REVIEWS.ISBN WHERE STOCK\_QUANTITY>0);

SELECT \* FROM CUSTOMER\_ORDER\_DETAILS;



* 1. Creating a View containing Books in Stock:

CREATE VIEW INVENTORY AS (SELECT BOOKS.TITLE AS TITLE, BOOKS.AUTHOR AS AUTHOR, BOOKS.GENRE AS GENRE, BOOKS.PUBLICATION\_YEAR AS PUBLICATION\_YEAR, BOOKS.PRICE AS PRICE, REVIEWS.RATING AS RATING, BOOKS.STOCK\_QUANTITY AS STOCK\_QUANTITY FROM BOOKS JOIN REVIEWS ON BOOKS.ISBN = REVIEWS.ISBN WHERE STOCK\_QUANTITY>0);

SELECT \* FROM INVENTORY;

