The Reading Loft

By: Marilyn Pamias

Summary of business:

The Reading Loft is a bookstore that operates both online and in a physical location. It offers a wide range of books across various categories, sourced from multiple publishers. The bookstore serves a large customer base. The bookstore problem is its need to manage a large amount of data related to its operations as the bookstore is growing in popularity. This data includes information about the books, orders, invoices, customers, employees, publishers, and more. This data needs to be organized for efficient operations and decision-making. The bookstore also needs to track inventory levels to ensure that it can meet customer demand while minimizing storage costs. The solution involves a well-designed database that organizes the bookstore's data into twelve entities that relate to each other to reflect the business process of the bookstore and address the issues the bookstore faces. The database will also provide a way to securely access data.

Problems and Constraints

The problems lie in inventory management, order processing, customer relationship management, and sales tracking. Inventory will need to be kept track of which includes the quantity and variety of books in stock. Having good management of the inventory will be useful to know when to reorder books, how many to order and to better manage storage space and its costs. Order processing is crucial as each order needs to be tracked, ensuring the ordered books are in stock and the delivery is managed effectively. Customer relationship management is essential to maintain a good relationship with customers, for this customer information must be managed. Sales tracking is important for business decisions, this includes tracking sales by book, category, and sales channel.

The constraints involve data consistency as the data across different tables needs to be consistent. For example, the total quantity of a book in the inventory table should match the sum of quantities in the order detail and cart information tables. Data privacy is another constraint as customer information needs to be handled securely and in compliance with privacy laws. As the amount of data grows maintaining a high level of performance can become challenging, making this another constraint.

The objectives therefore involve efficient operations between day-to-day operations and order processing and inventory management. The database should be able to provide valuable information for business decisions and can be expanded to other aspects like identifying best-selling books and understanding sales trends. The database's objective also involves being able to handle growth in terms of data volume and evolving business requirements. This will be achieved by implementing twelve entities that have the appropriate relationships to be able to realize these objectives in an organized manner.

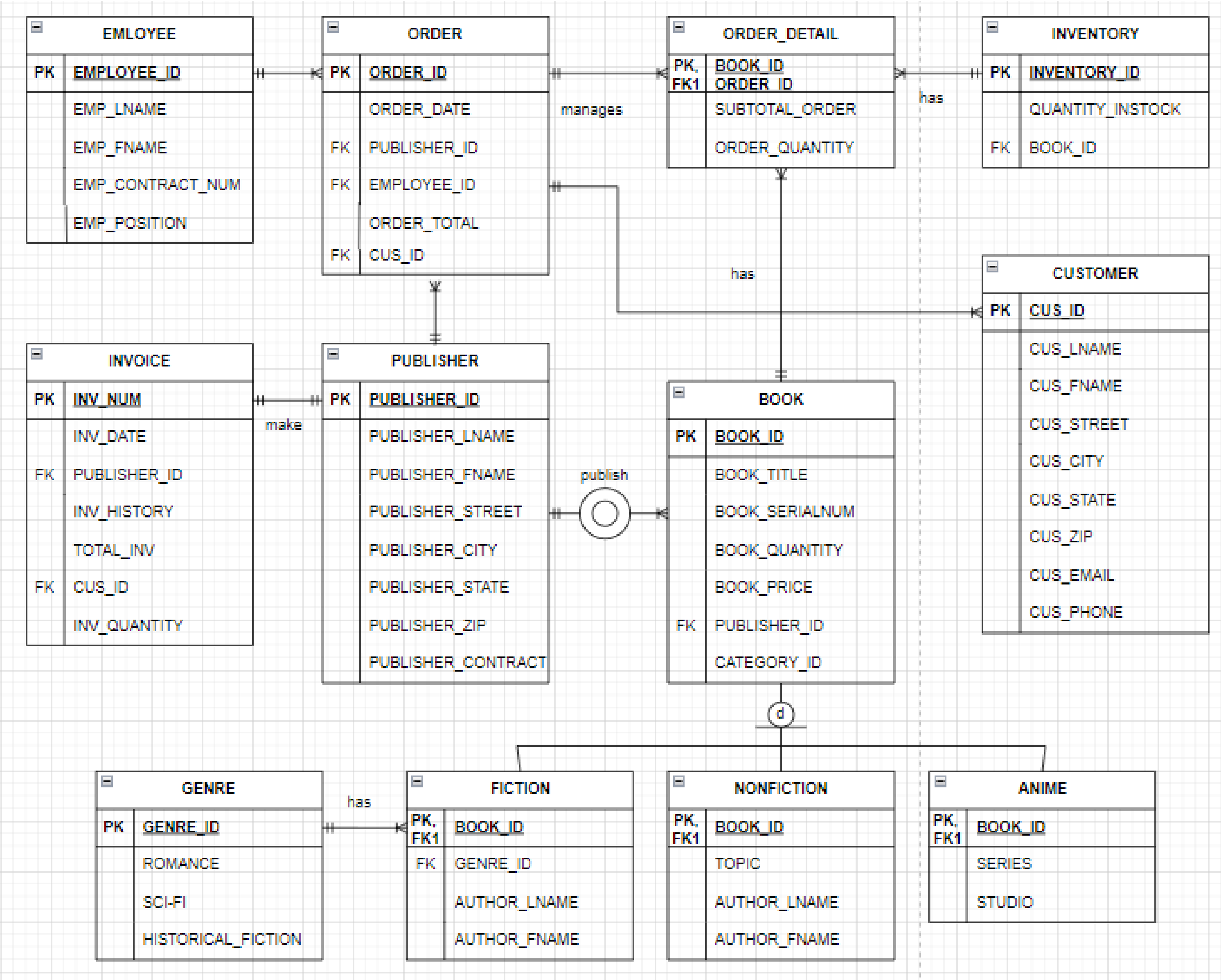
Scope and Boundaries

The scope of the database includes the following twelve entities which manage data related to: invoice, employee, inventory, customer, order, order detail, publisher, book, genre, fiction, nonfiction, and anime. The product is books, and the subtypes are fiction, nonfiction, and anime. The vendor is the publisher and invoices pertain to the publisher, while the order pertains to the customer. The relationships represent the processes of the bookstore and involve one to many and one to one relationship.

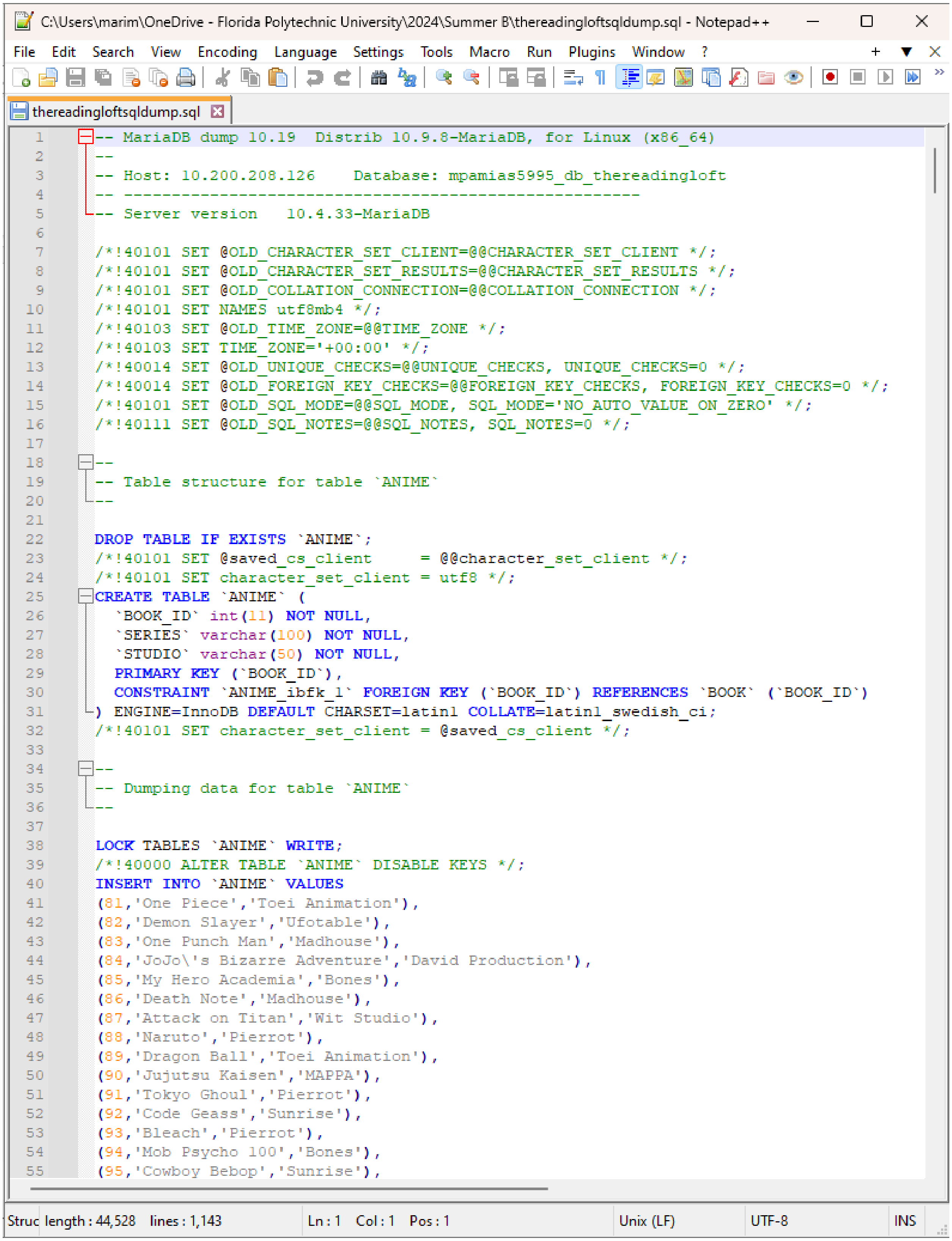
The boundaries of the bookstore database ensure the privacy and security of customer information, which complies with privacy laws and regulations. The database will maintain data consistency across all tables to ensure the information is accurate and up to date. It will also be able to accommodate growth of the bookstore and will be designed to handle a growing volume of data while maintaining high performance. The business rules are as follows:

- A publisher can have only one invoice, invoice is made by the publisher by the end of the month.
- A publisher can have one or many orders.
- A publisher can publish one or many books.
- An order can be associated with one or many order details.
- An employee can manage one or many orders.
- A category can be associated with many books, but a book can only have one category.
- A book can be associated with one or many order details.
- A book can only have one category.
- A customer can have one or many orders.
- A book can also be associated with many, or one order details.
- Each order is associated with many order details.
- Each inventory item can be associated with many order details.
- Each fiction book has one genre.
- Customer_ID can only be associated to single member and there shall be no null.

ERD

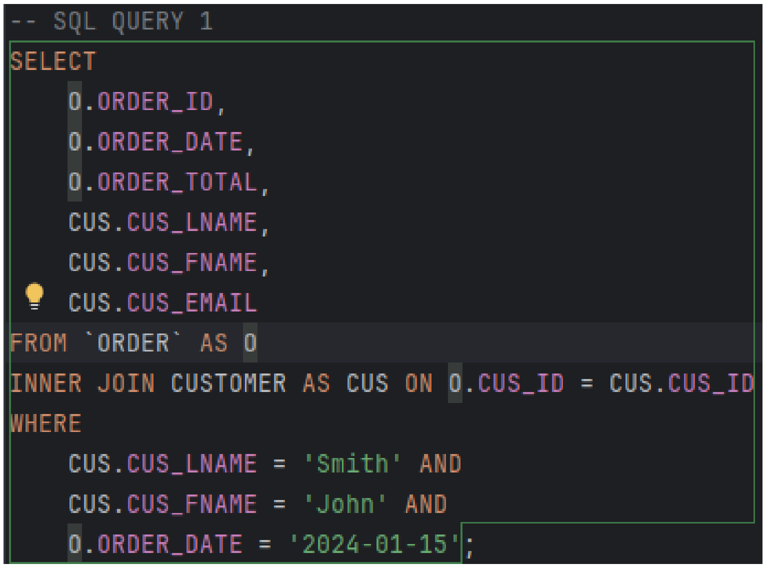

EMLOYEE
PK EMPLOYEE_ID
EMP_LNAME
EMP_FNAME
EMP_CONTRACT_NUM
EMP_POSITION
ORDER
PK ORDER_ID
ORDER_DATE
FK PUBLISHER_ID
FK EMPLOYEE_ID
ORDER_TOTAL
FK CUS_ID
manages
ORDER_DETAIL
PK, FK1 BOOK_ID
ORDER_ID
SUBTOTAL_ORDER
ORDER_QUANTITY
has
INVENTORY
PK INVENTORY_ID
QUANTITY_INSTOCK
FK BOOK_ID
has
CUSTOMER
PK CUS_ID
CUS_LNAME
CUS_FNAME
CUS_STREET
CUS_CITY
CUS_STATE
CUS_ZIP
CUS_EMAIL
CUS_PHONE
INVOICE
PK INV_NUM
INV_DATE
FK PUBLISHER_ID
INV_HISTORY
TOTAL_INV
FK CUS_ID
INV_QUANTITY
make
PUBLISHER
PK PUBLISHER_ID
PUBLISHER_LNAME
PUBLISHER_FNAME
PUBLISHER_STREET
PUBLISHER_CITY
PUBLISHER_STATE
PUBLISHER_ZIP
PUBLISHER_CONTRACT
publish
BOOK
PK BOOK_ID
BOOK_TITLE
BOOK_SERIALNUM
BOOK_QUANTITY
BOOK_PRICE
FK PUBLISHER_ID
CATEGORY_ID
d
GENRE
PK GENRE_ID
ROMANCE
SCI-FI
HISTORICAL_FICTION
has
FICTION
PK, FK1 BOOK_ID
FK GENRE_ID
AUTHOR_LNAME
AUTHOR_FNAME
NONFICTION
PK, FK1 BOOK_ID
TOPIC
AUTHOR_LNAME
AUTHOR_FNAME
ANIME
PK, FK1 BOOK_ID
SERIES
STUDIO

SQL Dump

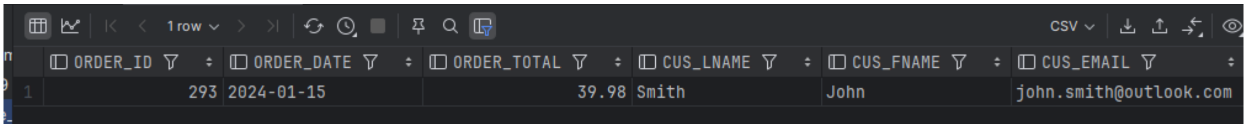


```sql
-- MariaDB dump 10.19  Distrib 10.9.8-MariaDB, for Linux (x86_64)
--
-- Host: 10.200.208.126    Database: mpamias5995_db_thereadingloft
-- ------------------------------------------------------
-- Server version	10.4.33-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `ANIME`
--

DROP TABLE IF EXISTS `ANIME`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `ANIME` (
  `BOOK_ID` int(11) NOT NULL,
  `SERIES` varchar(100) NOT NULL,
  `STUDIO` varchar(50) NOT NULL,
  PRIMARY KEY (`BOOK_ID`),
  CONSTRAINT `ANIME_ibfk_1` FOREIGN KEY (`BOOK_ID`) REFERENCES `BOOK` (`BOOK_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `ANIME`
--

LOCK TABLES `ANIME` WRITE;
/*!40000 ALTER TABLE `ANIME` DISABLE KEYS */;
INSERT INTO `ANIME` VALUES
(81,'One Piece','Toei Animation'),
(82,'Demon Slayer','Ufotable'),
(83,'One Punch Man','Madhouse'),
(84,'JoJo\'s Bizarre Adventure','David Production'),
(85,'My Hero Academia','Bones'),
(86,'Death Note','Madhouse'),
(87,'Attack on Titan','Wit Studio'),
(88,'Naruto','Pierrot'),
(89,'Dragon Ball','Toei Animation'),
(90,'Jujutsu Kaisen','MAPPA'),
(91,'Tokyo Ghoul','Pierrot'),
(92,'Code Geass','Sunrise'),
(93,'Bleach','Pierrot'),
(94,'Mob Psycho 100','Bones'),
(95,'Cowboy Bebop','Sunrise'),
```

SQL Query 1: Show a specific invoice based on a customer name and date of purchase.

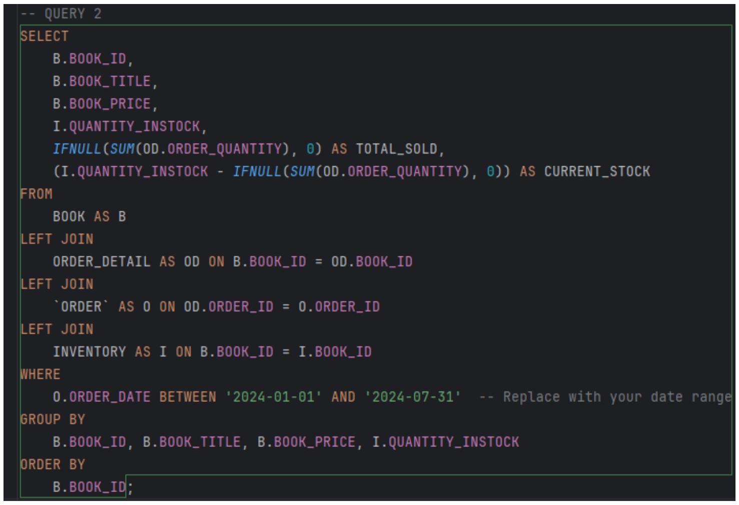


```
-- SQL QUERY 1
SELECT
    O.ORDER_ID,
    O.ORDER_DATE,
    O.ORDER_TOTAL,
    CUS.CUS_LNAME,
    CUS.CUS_FNAME,
    CUS.CUS_EMAIL
FROM `ORDER` AS O
INNER JOIN CUSTOMER AS CUS ON O.CUS_ID = CUS.CUS_ID
WHERE
    CUS.CUS_LNAME = 'Smith' AND
    CUS.CUS_FNAME = 'John' AND
    O.ORDER_DATE = '2024-01-15';
```

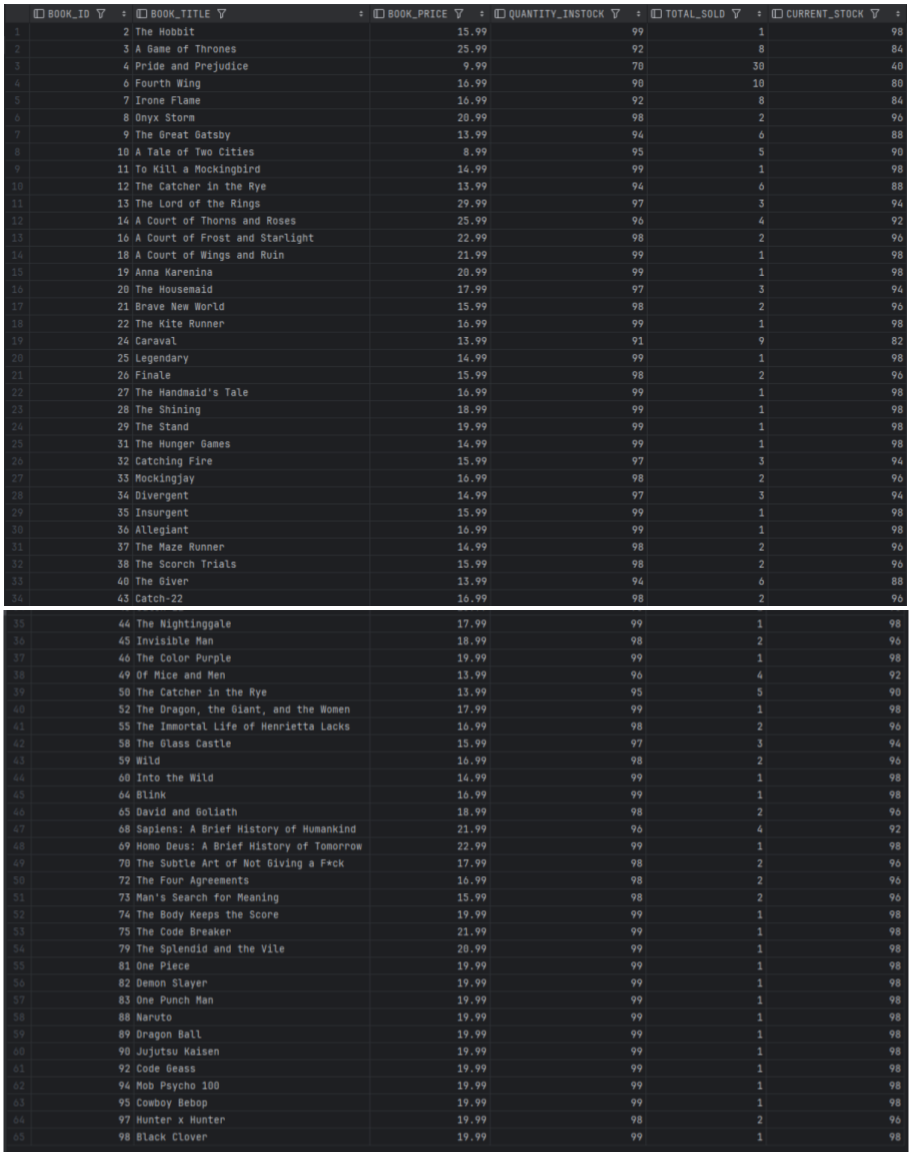


	ORDER_ID	ORDER_DATE	ORDER_TOTAL	CUS_LNAME	CUS_FNAME	CUS_EMAIL
1	293	2024-01-15	39.98	Smith	John	john.smith@outlook.com

SQL Query 2: Show a current inventory of all products sold within the date range.

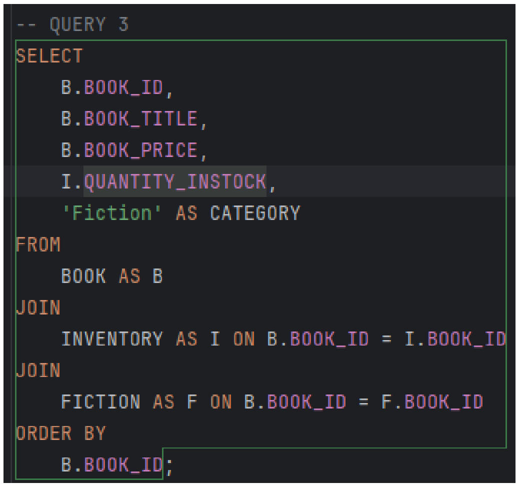


```sql
-- QUERY 2
SELECT
    B.BOOK_ID,
    B.BOOK_TITLE,
    B.BOOK_PRICE,
    I.QUANTITY_INSTOCK,
    IFNULL(SUM(OD.ORDER_QUANTITY), 0) AS TOTAL_SOLD,
    (I.QUANTITY_INSTOCK - IFNULL(SUM(OD.ORDER_QUANTITY), 0)) AS CURRENT_STOCK
FROM
    BOOK AS B
LEFT JOIN
    ORDER_DETAIL AS OD ON B.BOOK_ID = OD.BOOK_ID
LEFT JOIN
    `ORDER` AS O ON OD.ORDER_ID = O.ORDER_ID
LEFT JOIN
    INVENTORY AS I ON B.BOOK_ID = I.BOOK_ID
WHERE
    O.ORDER_DATE BETWEEN '2024-01-01' AND '2024-07-31'  -- Replace with your date range
GROUP BY
    B.BOOK_ID, B.BOOK_TITLE, B.BOOK_PRICE, I.QUANTITY_INSTOCK
ORDER BY
    B.BOOK_ID;
```

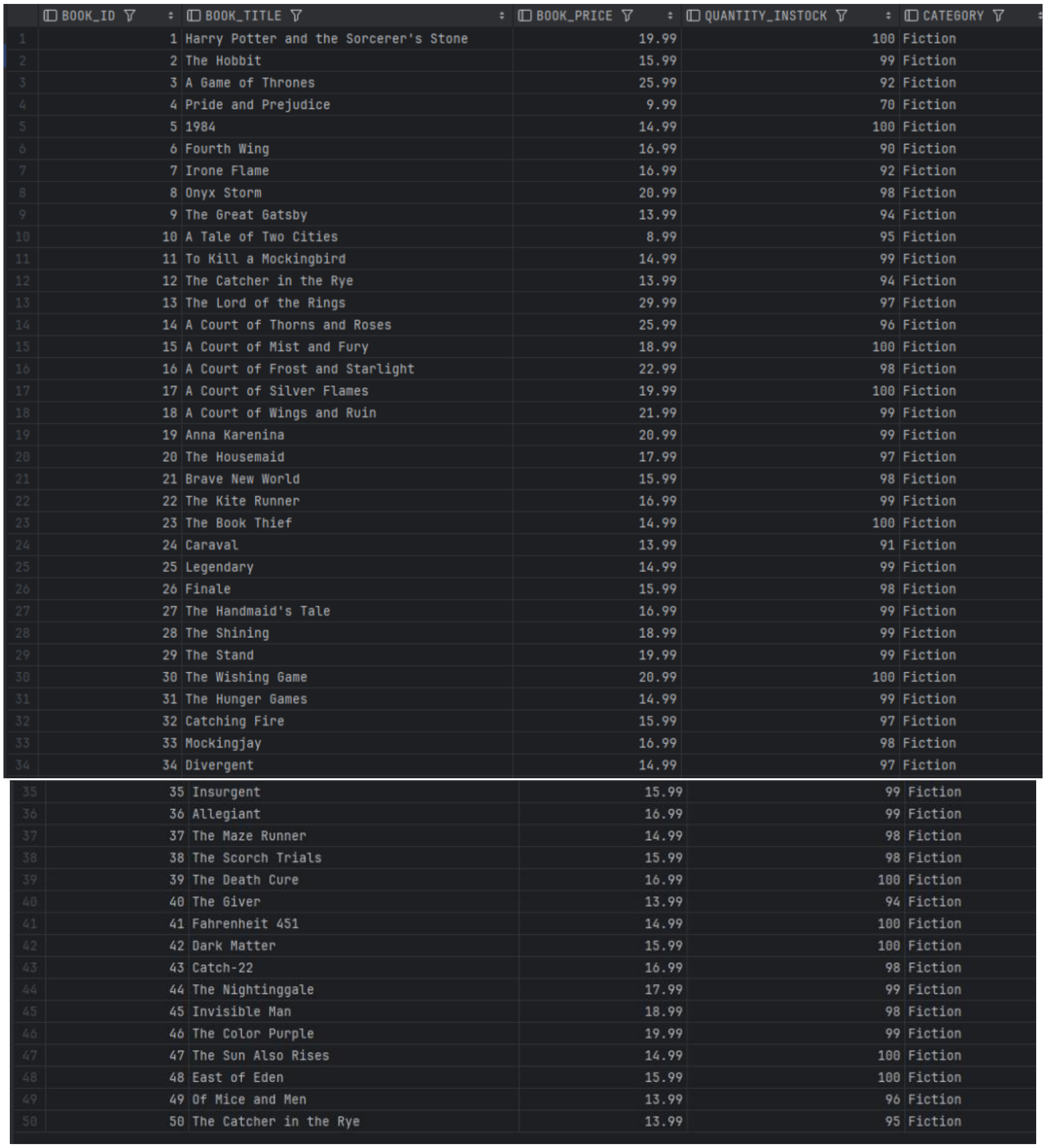


BOOK_ID	BOOK_TITLE	BOOK_PRICE	QUANTITY_INSTOCK	TOTAL_SOLD	CURRENT_STOCK
2	The Hobbit	15.99	99	1	98
3	A Game of Thrones	25.99	92	8	84
4	Pride and Prejudice	9.99	70	30	40
6	Fourth Wing	16.99	90	10	80
7	Irone Flame	16.99	92	8	84
8	Onyx Storm	20.99	98	2	96
9	The Great Gatsby	13.99	94	6	88
10	A Tale of Two Cities	8.99	95	5	90
11	To Kill a Mockingbird	14.99	99	1	98
12	The Catcher in the Rye	13.99	94	6	88
13	The Lord of the Rings	29.99	97	3	94
14	A Court of Thorns and Roses	25.99	96	4	92
16	A Court of Frost and Starlight	22.99	98	2	96
18	A Court of Wings and Ruin	21.99	99	1	98
19	Anna Karenina	20.99	99	1	98
20	The Housemaid	17.99	97	3	94
21	Brave New World	15.99	98	2	96
22	The Kite Runner	16.99	99	1	98
24	Caraval	13.99	91	9	82
25	Legendary	14.99	99	1	98
26	Finale	15.99	98	2	96
27	The Handmaid's Tale	16.99	99	1	98
28	The Shining	18.99	99	1	98
29	The Stand	19.99	99	1	98
31	The Hunger Games	14.99	99	1	98
32	Catching Fire	15.99	97	3	94
33	Mockingjay	16.99	98	2	96
34	Divergent	14.99	97	3	94
35	Insurgent	15.99	99	1	98
36	Allegiant	16.99	99	1	98
37	The Maze Runner	14.99	98	2	96
38	The Scorch Trials	15.99	98	2	96
40	The Giver	13.99	94	6	88
43	Catch-22	16.99	98	2	96
44	The Nightinggale	17.99	99	1	98
45	Invisible Man	18.99	98	2	96
46	The Color Purple	19.99	99	1	98
49	Of Mice and Men	13.99	96	4	92
50	The Catcher in the Rye	13.99	95	5	90
52	The Dragon, the Giant, and the Women	17.99	99	1	98
55	The Immortal Life of Henrietta Lacks	16.99	98	2	96
58	The Glass Castle	15.99	97	3	94
59	Wild	16.99	98	2	96
60	Into the Wild	14.99	99	1	98
64	Blink	16.99	99	1	98
65	David and Goliath	18.99	98	2	96
68	Sapiens: A Brief History of Humankind	21.99	96	4	92
69	Homo Deus: A Brief History of Tomorrow	22.99	99	1	98
70	The Subtle Art of Not Giving a F*ck	17.99	98	2	96
72	The Four Agreements	16.99	98	2	96
73	Man's Search for Meaning	15.99	98	2	96
74	The Body Keeps the Score	19.99	99	1	98
75	The Code Breaker	21.99	99	1	98
79	The Splendid and the Vile	20.99	99	1	98
81	One Piece	19.99	99	1	98
82	Demon Slayer	19.99	99	1	98
83	One Punch Man	19.99	99	1	98
88	Naruto	19.99	99	1	98
89	Dragon Ball	19.99	99	1	98
90	Jujutsu Kaisen	19.99	99	1	98
92	Code Geass	19.99	99	1	98
94	Mob Psycho 100	19.99	99	1	98
95	Cowboy Bebop	19.99	99	1	98
97	Hunter x Hunter	19.99	98	2	96
98	Black Clover	19.99	99	1	98

SQL Query 3: Show a current inventory of all products that match one specific product type.

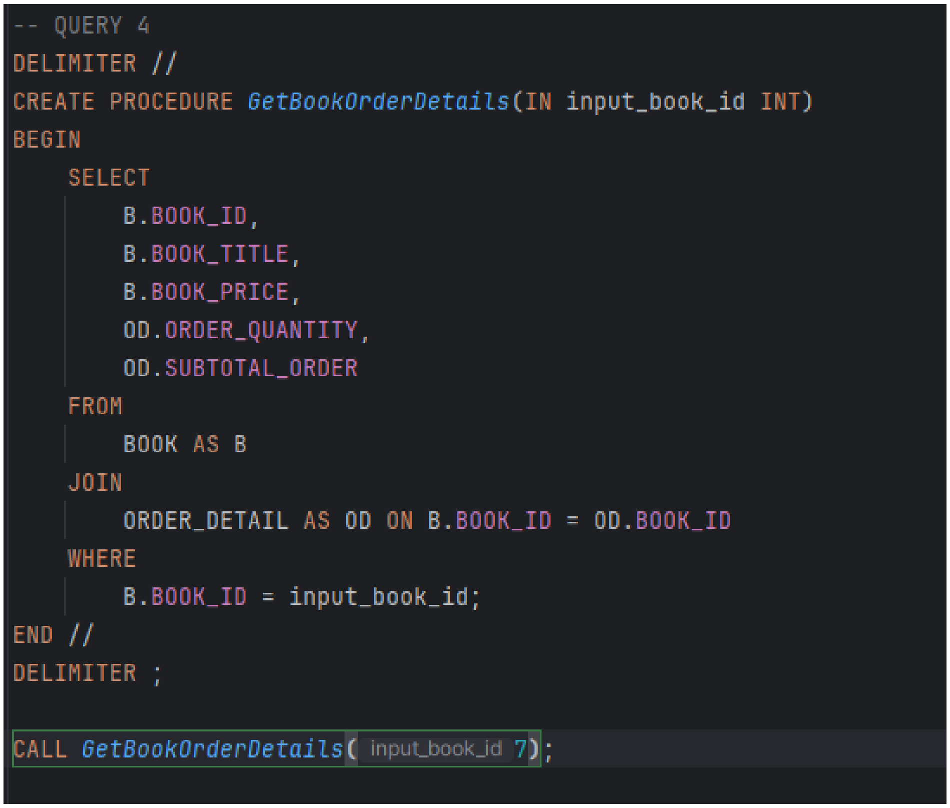


```sql
-- QUERY 3
SELECT
    B.BOOK_ID,
    B.BOOK_TITLE,
    B.BOOK_PRICE,
    I.QUANTITY_INSTOCK,
    'Fiction' AS CATEGORY
FROM
    BOOK AS B
JOIN
    INVENTORY AS I ON B.BOOK_ID = I.BOOK_ID
JOIN
    FICTION AS F ON B.BOOK_ID = F.BOOK_ID
ORDER BY
    B.BOOK_ID;
```

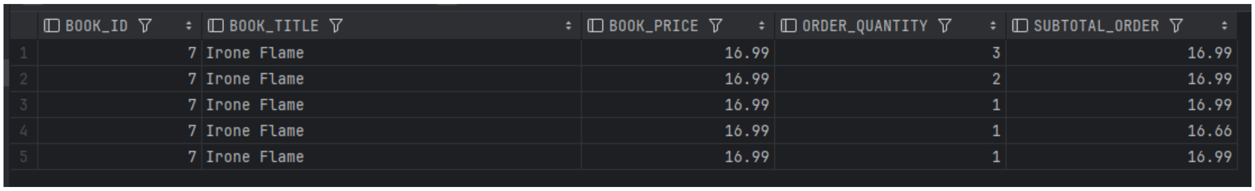


BOOK_ID	BOOK_TITLE	BOOK_PRICE	QUANTITY_INSTOCK	CATEGORY
1	Harry Potter and the Sorcerer's Stone	19.99	100	Fiction
2	The Hobbit	15.99	99	Fiction
3	A Game of Thrones	25.99	92	Fiction
4	Pride and Prejudice	9.99	70	Fiction
5	1984	14.99	100	Fiction
6	Fourth Wing	16.99	90	Fiction
7	Irone Flame	16.99	92	Fiction
8	Onyx Storm	20.99	98	Fiction
9	The Great Gatsby	13.99	94	Fiction
10	A Tale of Two Cities	8.99	95	Fiction
11	To Kill a Mockingbird	14.99	99	Fiction
12	The Catcher in the Rye	13.99	94	Fiction
13	The Lord of the Rings	29.99	97	Fiction
14	A Court of Thorns and Roses	25.99	96	Fiction
15	A Court of Mist and Fury	18.99	100	Fiction
16	A Court of Frost and Starlight	22.99	98	Fiction
17	A Court of Silver Flames	19.99	100	Fiction
18	A Court of Wings and Ruin	21.99	99	Fiction
19	Anna Karenina	20.99	99	Fiction
20	The Housemaid	17.99	97	Fiction
21	Brave New World	15.99	98	Fiction
22	The Kite Runner	16.99	99	Fiction
23	The Book Thief	14.99	100	Fiction
24	Caraval	13.99	91	Fiction
25	Legendary	14.99	99	Fiction
26	Finale	15.99	98	Fiction
27	The Handmaid's Tale	16.99	99	Fiction
28	The Shining	18.99	99	Fiction
29	The Stand	19.99	99	Fiction
30	The Wishing Game	20.99	100	Fiction
31	The Hunger Games	14.99	99	Fiction
32	Catching Fire	15.99	97	Fiction
33	Mockingjay	16.99	98	Fiction
34	Divergent	14.99	97	Fiction
35	Insurgent	15.99	99	Fiction
36	Allegiant	16.99	99	Fiction
37	The Maze Runner	14.99	98	Fiction
38	The Scorch Trials	15.99	98	Fiction
39	The Death Cure	16.99	100	Fiction
40	The Giver	13.99	94	Fiction
41	Fahrenheit 451	14.99	100	Fiction
42	Dark Matter	15.99	100	Fiction
43	Catch-22	16.99	98	Fiction
44	The Nightinggale	17.99	99	Fiction
45	Invisible Man	18.99	98	Fiction
46	The Color Purple	19.99	99	Fiction
47	The Sun Also Rises	14.99	100	Fiction
48	East of Eden	15.99	100	Fiction
49	Of Mice and Men	13.99	96	Fiction
50	The Catcher in the Rye	13.99	95	Fiction

SQL Query 4: SQL stored procedure that allows user to pass in a value and in order to join two tables to produce an output (you decide the attributes).

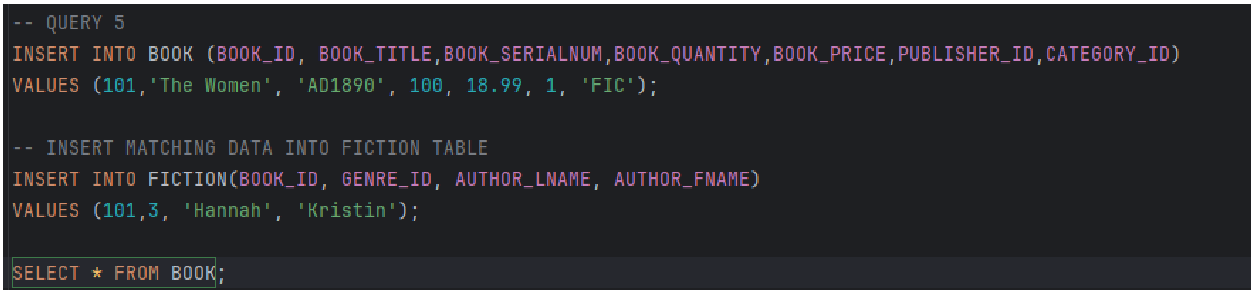


```
-- QUERY 4
DELIMITER //
CREATE PROCEDURE GetBookOrderDetails(IN input_book_id INT)
BEGIN
    SELECT
        B.BOOK_ID,
        B.BOOK_TITLE,
        B.BOOK_PRICE,
        OD.ORDER_QUANTITY,
        OD.SUBTOTAL_ORDER
    FROM
        BOOK AS B
    JOIN
        ORDER_DETAIL AS OD ON B.BOOK_ID = OD.BOOK_ID
    WHERE
        B.BOOK_ID = input_book_id;
END //
DELIMITER ;

CALL GetBookOrderDetails( input_book_id 7);
```

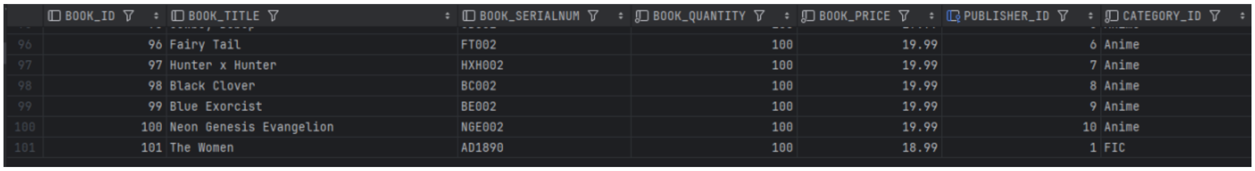


	BOOK_ID	BOOK_TITLE	BOOK_PRICE	ORDER_QUANTITY	SUBTOTAL_ORDER
1	7	Irone Flame	16.99	3	16.99
2	7	Irone Flame	16.99	2	16.99
3	7	Irone Flame	16.99	1	16.99
4	7	Irone Flame	16.99	1	16.66
5	7	Irone Flame	16.99	1	16.99

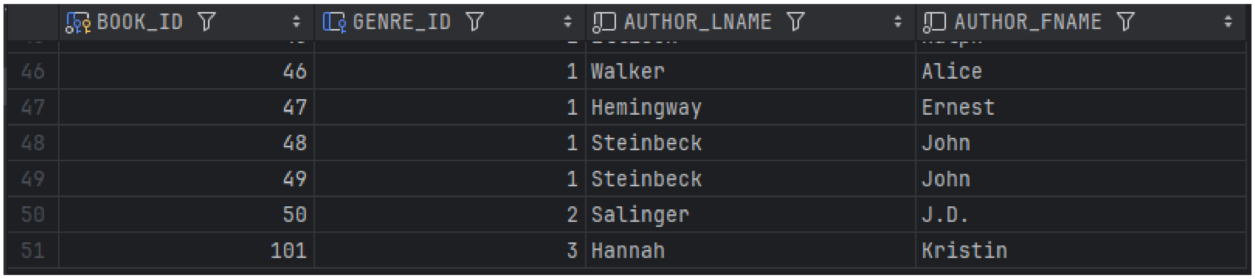
SQL Query 5: SQL code used to insert new product into product table (show all attributes).



```
-- QUERY 5
INSERT INTO BOOK (BOOK_ID, BOOK_TITLE,BOOK_SERIALNUM,BOOK_QUANTITY,BOOK_PRICE,PUBLISHER_ID,CATEGORY_ID)
VALUES (101,'The Women', 'AD1890', 100, 18.99, 1, 'FIC');

-- INSERT MATCHING DATA INTO FICTION TABLE
INSERT INTO FICTION(BOOK_ID, GENRE_ID, AUTHOR_LNAME, AUTHOR_FNAME)
VALUES (101,3, 'Hannah', 'Kristin');

SELECT * FROM BOOK;
```

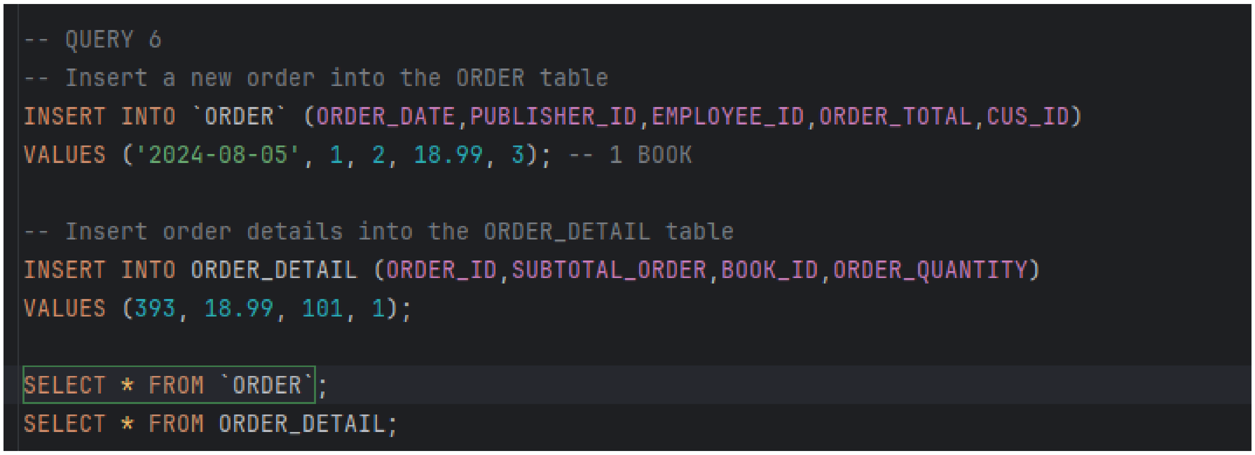


	BOOK_ID	BOOK_TITLE	BOOK_SERIALNUM	BOOK_QUANTITY	BOOK_PRICE	PUBLISHER_ID	CATEGORY_ID
96	96	Fairy Tail	FT002	100	19.99	6	Anime
97	97	Hunter x Hunter	HXH002	100	19.99	7	Anime
98	98	Black Clover	BC002	100	19.99	8	Anime
99	99	Blue Exorcist	BE002	100	19.99	9	Anime
100	100	Neon Genesis Evangelion	NGE002	100	19.99	10	Anime
101	101	The Women	AD1890	100	18.99	1	FIC

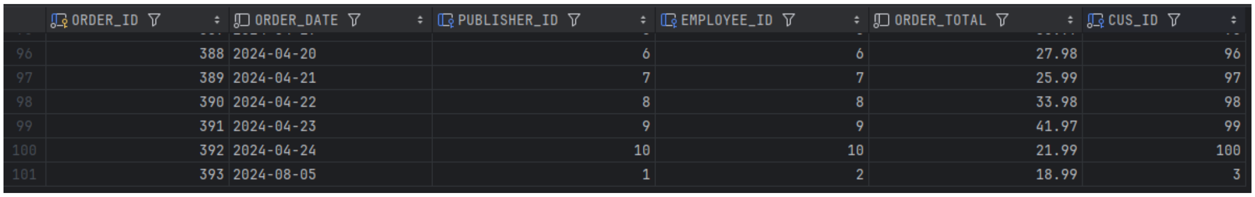


	BOOK_ID	GENRE_ID	AUTHOR_LNAME	AUTHOR_FNAME
46	46	1	Walker	Alice
47	47	1	Hemingway	Ernest
48	48	1	Steinbeck	John
49	49	1	Steinbeck	John
50	50	2	Salinger	J.D.
51	101	3	Hannah	Kristin

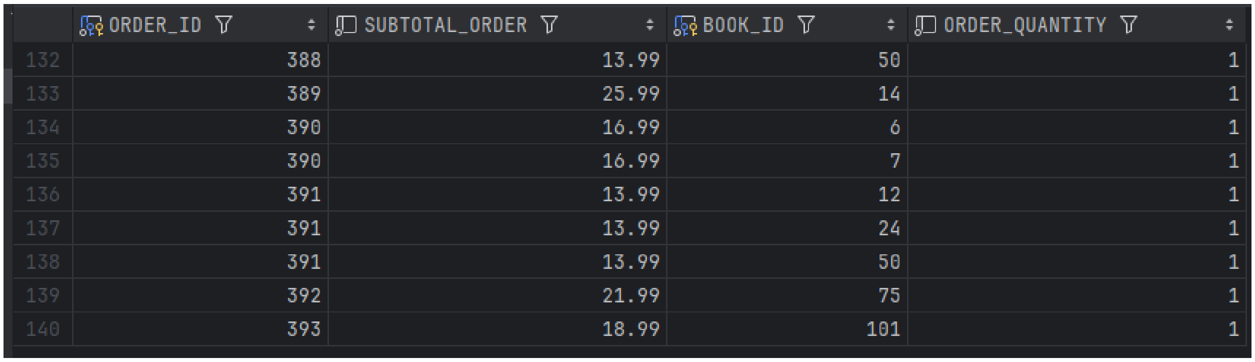
SQL Query 6: SQL code to generate a new invoice, you decide product and customer information.



```
-- QUERY 6
-- Insert a new order into the ORDER table
INSERT INTO `ORDER` (ORDER_DATE,PUBLISHER_ID,EMPLOYEE_ID,ORDER_TOTAL,CUS_ID)
VALUES ('2024-08-05', 1, 2, 18.99, 3); -- 1 BOOK

-- Insert order details into the ORDER_DETAIL table
INSERT INTO ORDER_DETAIL (ORDER_ID,SUBTOTAL_ORDER,BOOK_ID,ORDER_QUANTITY)
VALUES (393, 18.99, 101, 1);

SELECT * FROM `ORDER`;
SELECT * FROM ORDER_DETAIL;
```



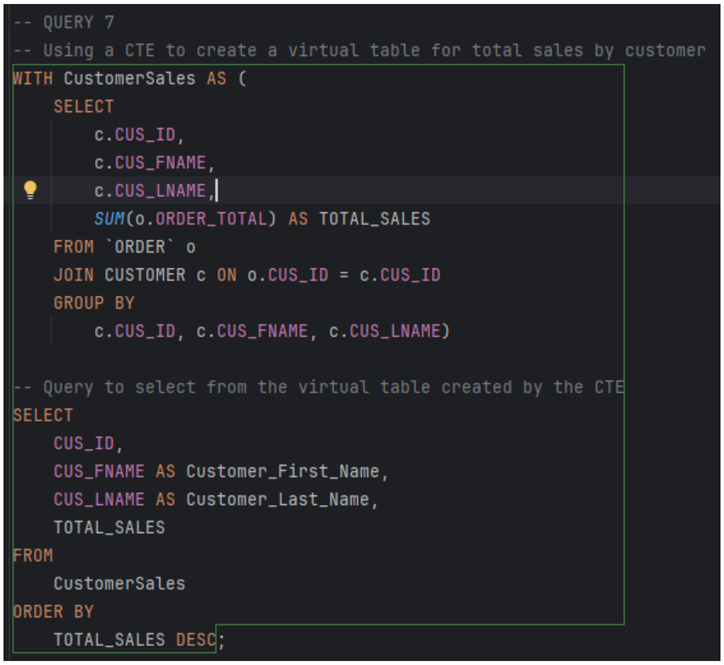
	ORDER_ID	ORDER_DATE	PUBLISHER_ID	EMPLOYEE_ID	ORDER_TOTAL	CUS_ID
96	388	2024-04-20	6	6	27.98	96
97	389	2024-04-21	7	7	25.99	97
98	390	2024-04-22	8	8	33.98	98
99	391	2024-04-23	9	9	41.97	99
100	392	2024-04-24	10	10	21.99	100
101	393	2024-08-05	1	2	18.99	3



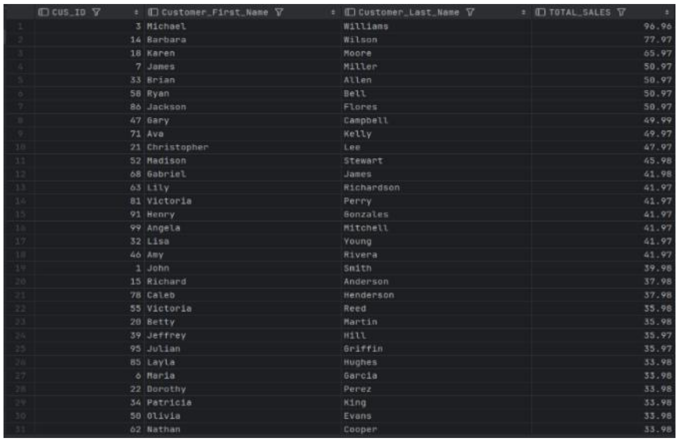
	ORDER_ID	SUBTOTAL_ORDER	BOOK_ID	ORDER_QUANTITY
132	388	13.99	50	1
133	389	25.99	14	1
134	390	16.99	6	1
135	390	16.99	7	1
136	391	13.99	12	1
137	391	13.99	24	1
138	391	13.99	50	1
139	392	21.99	75	1
140	393	18.99	101	1

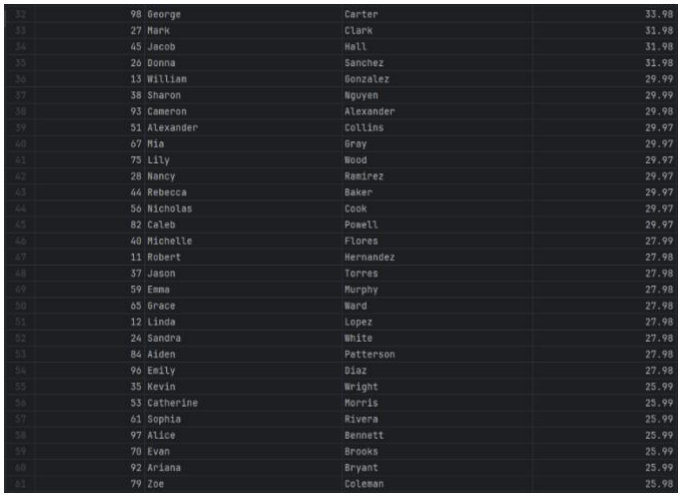
SQL Query 7:

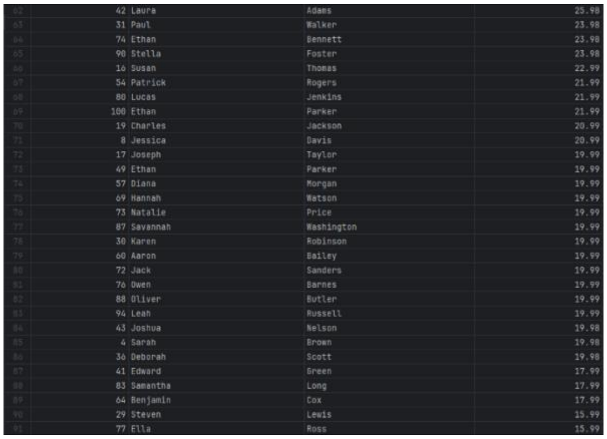
Show a virtual table (of your choice).

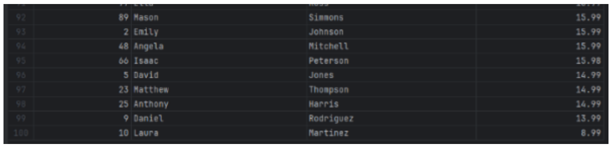


```sql
-- QUERY 7
-- Using a CTE to create a virtual table for total sales by customer
WITH CustomerSales AS (
    SELECT
        c.CUS_ID,
        c.CUS_FNAME,
        c.CUS_LNAME,
        SUM(o.ORDER_TOTAL) AS TOTAL_SALES
    FROM `ORDER` o
    JOIN CUSTOMER c ON o.CUS_ID = c.CUS_ID
    GROUP BY
        c.CUS_ID, c.CUS_FNAME, c.CUS_LNAME)

-- Query to select from the virtual table created by the CTE
SELECT
    CUS_ID,
    CUS_FNAME AS Customer_First_Name,
    CUS_LNAME AS Customer_Last_Name,
    TOTAL_SALES
FROM
    CustomerSales
ORDER BY
    TOTAL_SALES DESC;
```



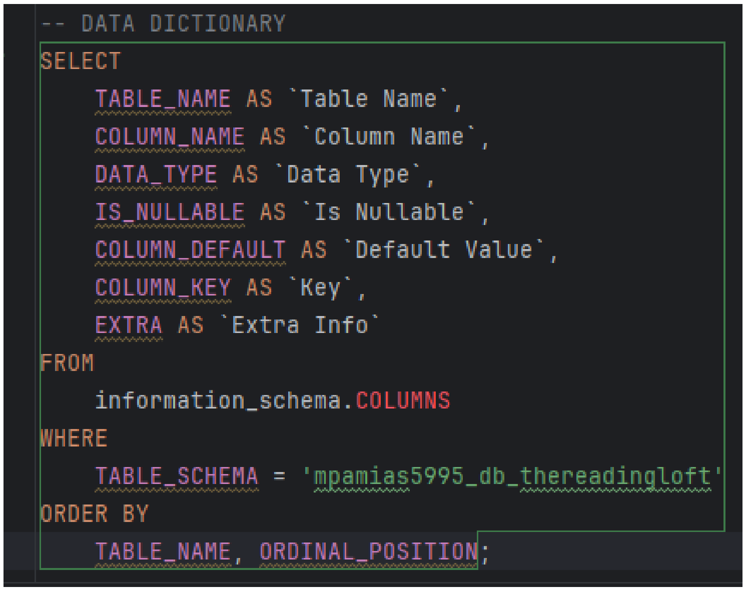




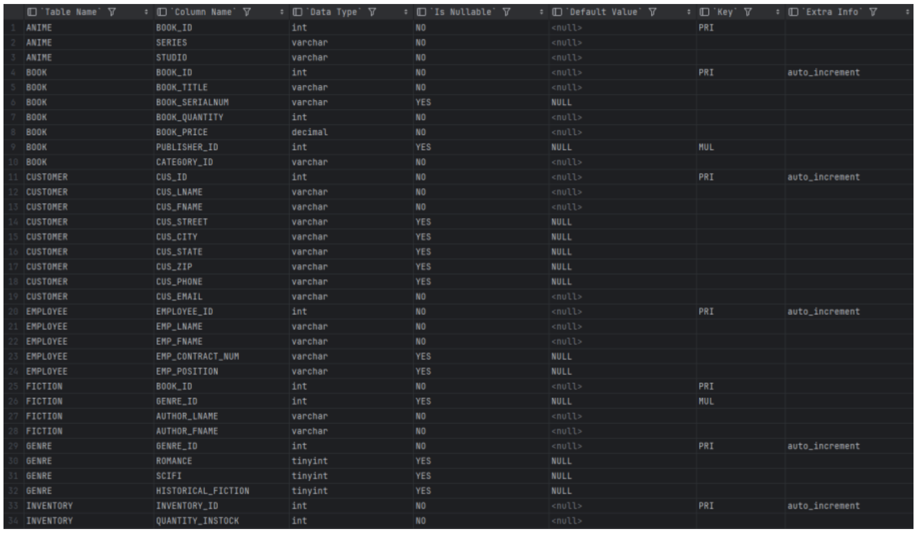


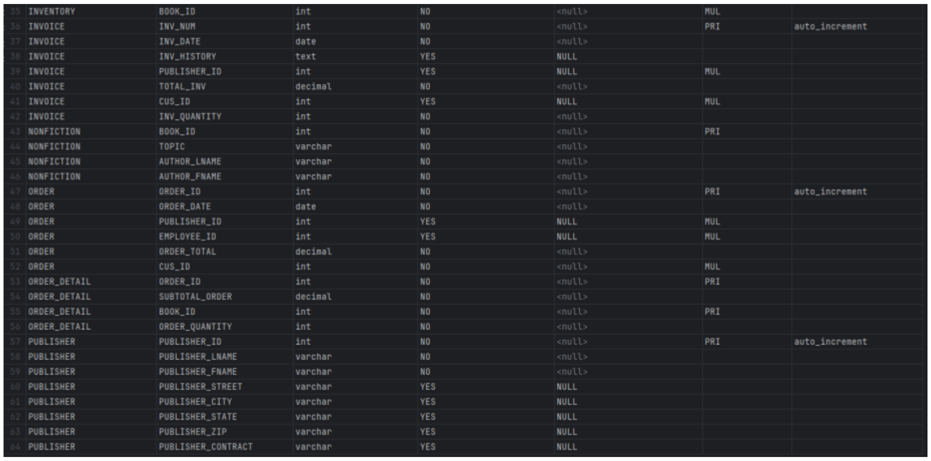
	CUS_ID	Customer_First_Name	Customer_Last_Name	TOTAL_SALES
1	3	Michael	Williams	96.96
2	14	Barbara	Wilson	77.97
3	18	Karen	Moore	65.97
4	7	James	Miller	50.97
5	33	Brian	Allen	50.97
6	58	Ryan	Bell	50.97
7	86	Jackson	Flores	50.97
8	47	Gary	Campbell	49.99
9	71	Ava	Kelly	49.97
10	21	Christopher	Lee	47.97
11	52	Madison	Stewart	45.98
12	68	Gabriel	James	41.98
13	63	Lily	Richardson	41.97
14	81	Victoria	Perry	41.97
15	91	Henry	Gonzales	41.97
16	99	Angela	Mitchell	41.97
17	32	Lisa	Young	41.97
18	46	Amy	Rivera	41.97
19	1	John	Smith	39.98
20	15	Richard	Anderson	37.98
21	78	Caleb	Henderson	37.98
22	55	Victoria	Reed	35.98
23	20	Betty	Martin	35.98
24	39	Jeffrey	Hill	35.97
25	95	Julian	Griffin	35.97
26	85	Layla	Hughes	33.98
27	6	Maria	Garcia	33.98
28	22	Dorothy	Perez	33.98
29	34	Patricia	King	33.98
30	50	Olivia	Evans	33.98
31	62	Nathan	Cooper	33.98
32	98	George	Carter	33.98
33	27	Mark	Clark	31.98
34	45	Jacob	Hall	31.98
35	26	Donna	Sanchez	31.98
36	13	William	Gonzalez	29.99
37	38	Sharon	Nguyen	29.99
38	93	Cameron	Alexander	29.98
39	51	Alexander	Collins	29.97
40	67	Mia	Gray	29.97
41	75	Lily	Wood	29.97
42	28	Nancy	Ramirez	29.97
43	44	Rebecca	Baker	29.97
44	56	Nicholas	Cook	29.97
45	82	Caleb	Powell	29.97
46	40	Michelle	Flores	27.99
47	11	Robert	Hernandez	27.98
48	37	Jason	Torres	27.98
49	59	Emma	Murphy	27.98
50	65	Grace	Ward	27.98
51	12	Linda	Lopez	27.98
52	24	Sandra	White	27.98
53	84	Aiden	Patterson	27.98
54	96	Emily	Diaz	27.98
55	35	Kevin	Wright	25.99
56	53	Catherine	Morris	25.99
57	61	Sophia	Rivera	25.99
58	97	Alice	Bennett	25.99
59	70	Evan	Brooks	25.99
60	92	Ariana	Bryant	25.99
61	79	Zoe	Coleman	25.98
62	42	Laura	Adams	25.98
63	31	Paul	Walker	23.98
64	74	Ethan	Bennett	23.98
65	90	Stella	Foster	23.98
66	16	Susan	Thomas	22.99
67	54	Patrick	Rogers	21.99
68	80	Lucas	Jenkins	21.99
69	100	Ethan	Parker	21.99
70	19	Charles	Jackson	20.99
71	8	Jessica	Davis	20.99
72	17	Joseph	Taylor	19.99
73	49	Ethan	Parker	19.99
74	57	Diana	Morgan	19.99
75	69	Hannah	Watson	19.99
76	73	Natalie	Price	19.99
77	87	Savannah	Washington	19.99
78	30	Karen	Robinson	19.99
79	60	Aaron	Bailey	19.99
80	72	Jack	Sanders	19.99
81	76	Owen	Barnes	19.99
82	88	Oliver	Butler	19.99
83	94	Leah	Russell	19.99
84	43	Joshua	Nelson	19.98
85	4	Sarah	Brown	19.98
86	36	Deborah	Scott	19.98
87	41	Edward	Green	17.99
88	83	Samantha	Long	17.99
89	64	Benjamin	Cox	17.99
90	29	Steven	Lewis	15.99
91	77	Ella	Ross	15.99
92	89	Mason	Simmons	15.99
93	2	Emily	Johnson	15.99
94	48	Angela	Mitchell	15.99
95	66	Isaac	Peterson	15.98
96	5	David	Jones	14.99
97	23	Matthew	Thompson	14.99
98	25	Anthony	Harris	14.99
99	9	Daniel	Rodriguez	13.99
100	10	Laura	Martinez	8.99

SQL Query 8: Show data dictionary of tables.



```sql
-- DATA DICTIONARY
SELECT
    TABLE_NAME AS `Table Name`,
    COLUMN_NAME AS `Column Name`,
    DATA_TYPE AS `Data Type`,
    IS_NULLABLE AS `Is Nullable`,
    COLUMN_DEFAULT AS `Default Value`,
    COLUMN_KEY AS `Key`,
    EXTRA AS `Extra Info`
FROM
    information_schema.COLUMNS
WHERE
    TABLE_SCHEMA = 'mpamias5995_db_thereadingloft'
ORDER BY
    TABLE_NAME, ORDINAL_POSITION;
```





	`Table Name`	`Column Name`	`Data Type`	`Is Nullable`	`Default Value`	`Key`	`Extra Info`
1	ANIME	BOOK_ID	int	NO	<null>	PRI	
2	ANIME	SERIES	varchar	NO	<null>		
3	ANIME	STUDIO	varchar	NO	<null>		
4	BOOK	BOOK_ID	int	NO	<null>	PRI	auto_increment
5	BOOK	BOOK_TITLE	varchar	NO	<null>		
6	BOOK	BOOK_SERIALNUM	varchar	YES	NULL		
7	BOOK	BOOK_QUANTITY	int	NO	<null>		
8	BOOK	BOOK_PRICE	decimal	NO	<null>		
9	BOOK	PUBLISHER_ID	int	YES	NULL	MUL	
10	BOOK	CATEGORY_ID	varchar	NO	<null>		
11	CUSTOMER	CUS_ID	int	NO	<null>	PRI	auto_increment
12	CUSTOMER	CUS_LNAME	varchar	NO	<null>		
13	CUSTOMER	CUS_FNAME	varchar	NO	<null>		
14	CUSTOMER	CUS_STREET	varchar	YES	NULL		
15	CUSTOMER	CUS_CITY	varchar	YES	NULL		
16	CUSTOMER	CUS_STATE	varchar	YES	NULL		
17	CUSTOMER	CUS_ZIP	varchar	YES	NULL		
18	CUSTOMER	CUS_PHONE	varchar	YES	NULL		
19	CUSTOMER	CUS_EMAIL	varchar	NO	<null>		
20	EMPLOYEE	EMPLOYEE_ID	int	NO	<null>	PRI	auto_increment
21	EMPLOYEE	EMP_LNAME	varchar	NO	<null>		
22	EMPLOYEE	EMP_FNAME	varchar	NO	<null>		
23	EMPLOYEE	EMP_CONTRACT_NUM	varchar	YES	NULL		
24	EMPLOYEE	EMP_POSITION	varchar	YES	NULL		
25	FICTION	BOOK_ID	int	NO	<null>	PRI	
26	FICTION	GENRE_ID	int	YES	NULL	MUL	
27	FICTION	AUTHOR_LNAME	varchar	NO	<null>		
28	FICTION	AUTHOR_FNAME	varchar	NO	<null>		
29	GENRE	GENRE_ID	int	NO	<null>	PRI	auto_increment
30	GENRE	ROMANCE	tinyint	YES	NULL		
31	GENRE	SCIFI	tinyint	YES	NULL		
32	GENRE	HISTORICAL_FICTION	tinyint	YES	NULL		
33	INVENTORY	INVENTORY_ID	int	NO	<null>	PRI	auto_increment
34	INVENTORY	QUANTITY_INSTOCK	int	NO	<null>		
35	INVENTORY	BOOK_ID	int	NO	<null>	MUL	
36	INVOICE	INV_NUM	int	NO	<null>	PRI	auto_increment
37	INVOICE	INV_DATE	date	NO	<null>		
38	INVOICE	INV_HISTORY	text	YES	NULL		
39	INVOICE	PUBLISHER_ID	int	YES	NULL	MUL	
40	INVOICE	TOTAL_INV	decimal	NO	<null>		
41	INVOICE	CUS_ID	int	YES	NULL	MUL	
42	INVOICE	INV_QUANTITY	int	NO	<null>		
43	NONFICTION	BOOK_ID	int	NO	<null>	PRI	
44	NONFICTION	TOPIC	varchar	NO	<null>		
45	NONFICTION	AUTHOR_LNAME	varchar	NO	<null>		
46	NONFICTION	AUTHOR_FNAME	varchar	NO	<null>		
47	ORDER	ORDER_ID	int	NO	<null>	PRI	auto_increment
48	ORDER	ORDER_DATE	date	NO	<null>		
49	ORDER	PUBLISHER_ID	int	YES	NULL	MUL	
50	ORDER	EMPLOYEE_ID	int	YES	NULL	MUL	
51	ORDER	ORDER_TOTAL	decimal	NO	<null>		
52	ORDER	CUS_ID	int	NO	<null>	MUL	
53	ORDER_DETAIL	ORDER_ID	int	NO	<null>	PRI	
54	ORDER_DETAIL	SUBTOTAL_ORDER	decimal	NO	<null>		
55	ORDER_DETAIL	BOOK_ID	int	NO	<null>	PRI	
56	ORDER_DETAIL	ORDER_QUANTITY	int	NO	<null>		
57	PUBLISHER	PUBLISHER_ID	int	NO	<null>	PRI	auto_increment
58	PUBLISHER	PUBLISHER_LNAME	varchar	NO	<null>		
59	PUBLISHER	PUBLISHER_FNAME	varchar	NO	<null>		
60	PUBLISHER	PUBLISHER_STREET	varchar	YES	NULL		
61	PUBLISHER	PUBLISHER_CITY	varchar	YES	NULL		
62	PUBLISHER	PUBLISHER_STATE	varchar	YES	NULL		
63	PUBLISHER	PUBLISHER_ZIP	varchar	YES	NULL		
64	PUBLISHER	PUBLISHER_CONTRACT	varchar	YES	NULL		

Database Solution Summary

Our database for The Reading Loft provides a robust and comprehensive framework that effectively manages the many aspects of the day-to-day business tasks. The solution is designed to support the efficient handling of employees, customers, orders, books, vendors, and inventory while ensuring data integrity and security as well as ease of use. The tables and relationships align with real-time management obligations with room for further data as the company continues to expand.

Triggers and automation were implemented to automatically update order totals when new orders are added, which helps with the financial aspect of business as well as determining which merchandise is sold the most as opposed to which is sold the least. Triggers were also implemented to update the inventory to maintain accurate stock records, and customer order totals also have a trigger to keep track of such. Dynamic queries were implemented and allow for easy and flexible retrieval of data based on the user input.

The database is made to be scalable, adaptable, and user-friendly which supports all business needs and ensure most accurate information to make effective business decisions. It provides a foundation for future growth and expansion and integrates various aspects of the business into a cohesive system which improves data management, operational efficiency, and decision-making capabilities.