

— Task 25 —

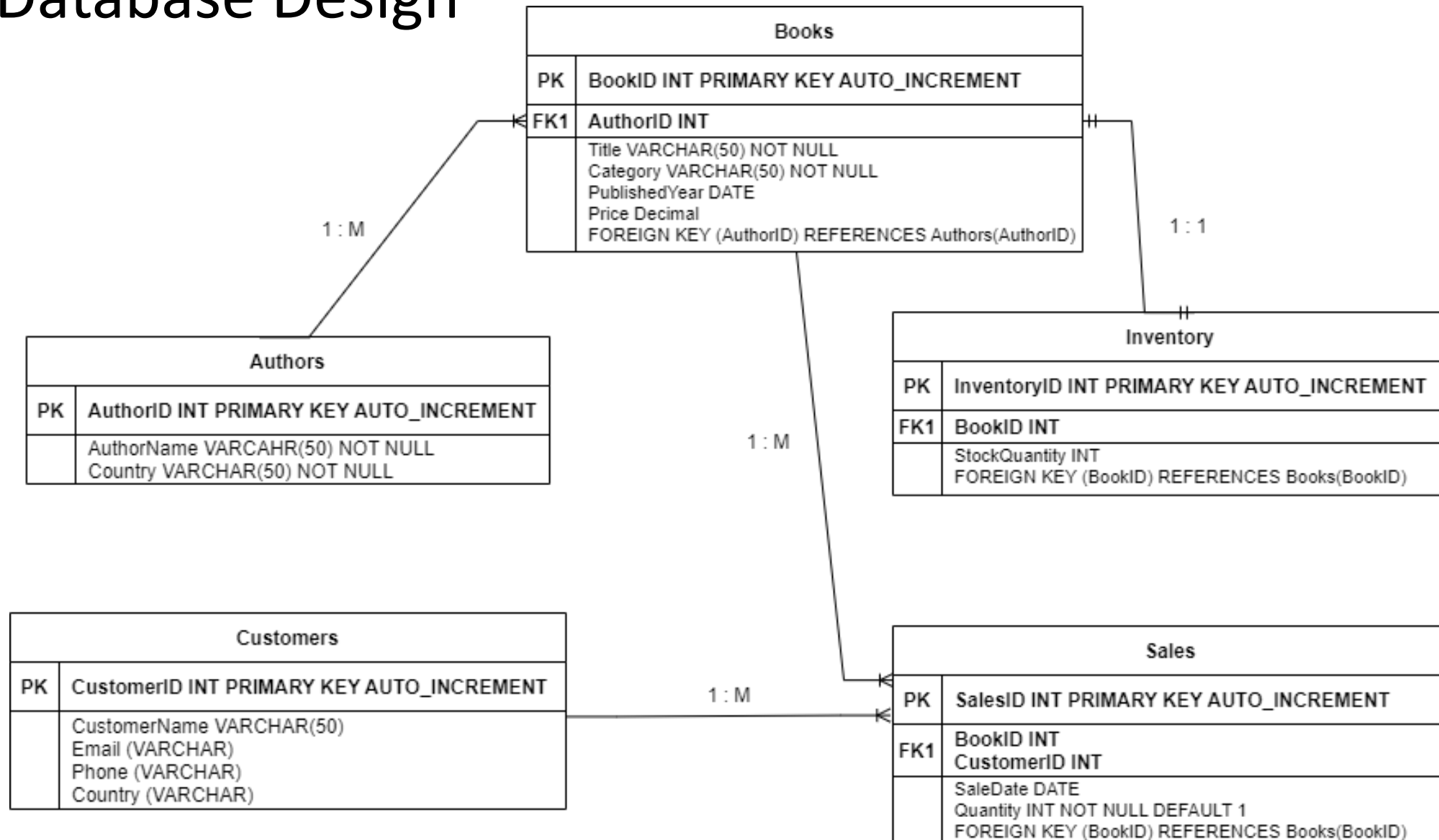
Analysis Sales Data

Name: Basel Amr Barakat
Email: baselamr52@gmail.com

Introduction

- Objective : Develop a comprehensive bookstore database to manage authors, books, and sales.
- Scope : Design tables for Authors, Books, and Sales; insert sample data; perform SQL queries.

Database Design



Requirement One

Create Database and Tables

1.1 Create Author Table

Table 1 : Authors

Col_Name	Type	Description
AuthorID	INT	to store the name of the author PRIMARY KEY
AuthorName	VARCHAR	to store the name of the author
Country	VARCHAR	to store the country of the author

We will Create the Table using the following command

```
CREATE TABLE IF NOT EXIST Authors(  
    AuthorID INT PRIMARY KEY AUTOINCREMENT,  
    AuthorName VARCHAR(50) NOT NULL,  
    Country VARCHAR(50) NOT NULL  
)
```

Authors	
PK	AuthorID INT PRIMARY KEY AUTO_INCREMENT
	AuthorName VARCHAR(50) NOT NULL Country VARCHAR(50) NOT NULL

```
1 # Create Authors Table  
2 cursor.execute('''  
3 CREATE TABLE IF NOT EXISTS Authors(  
4     AuthorID INTEGER PRIMARY KEY AUTOINCREMENT,  
5     AuthorName VARCHAR(50) NOT NULL,  
6     Country VARCHAR(50) NOT NULL  
7 );  
8 ''')  
9 conn.commit()  
10  
11 print("Authors table created successfully!")  
12 # Display Books table structure  
13 books_df = pd.read_sql_query("PRAGMA table_info(Authors);", conn)  
14 books_df
```

Authors table created successfully!

	cid	name	type	notnull	dflt_value	pk
0	0	AuthorID	INTEGER	0	None	1
1	1	AuthorName	VARCHAR(50)	1	None	0
2	2	Country	VARCHAR(50)	1	None	0

1.2 Create Books Table

Table 2 : Books

Col_Name	Type	Description
BookID	INT	to store the id of the book PRIMARY KEY
AuthorID	INT	to link between the book and it's author FOREIGN KEY
Title	VARCHAR	to store the name of the book
Category	VARCHAR	to store the category of the book
PublishedYear	DATE	to store the date of publication of the book
Price	DECIMAL(10,2)	to store the price of the book

We will Create the Table using the following command

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(50) NOT NULL,
    AuthorID INT,
    Category VARCHAR(50) NOT NULL,
    PublishedYear DATE,
    Price DECIMAL(10,2),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);
```

Books	
PK	BookID INT PRIMARY KEY AUTO_INCREMENT
FK1	AuthorID INT
	Title VARCHAR(50) NOT NULL Category VARCHAR(50) NOT NULL PublishedYear DATE Price Decimal FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)

```
[ ] 1 # Create Books Table
2 cursor.execute('''
3 CREATE TABLE IF NOT EXISTS Books(
4     BookID INTEGER PRIMARY KEY AUTOINCREMENT,
5     Title VARCHAR NOT NULL,
6     AuthorID INTEGER,
7     Category VARCHAR NOT NULL,
8     PublishedYear DATE,
9     Price DECIMAL,
10    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
11 );
12 ''')
13 conn.commit()
14
15 print("Books table created successfully!")
16
17 # Display Books table structure
18 books_df = pd.read_sql_query("PRAGMA table_info(Books);", conn)
19 books_df
20
```

Books table created successfully!

	cid	name	type	notnull	dflt_value	pk
0	0	BookID	INTEGER	0	None	1
1	1	Title	VARCHAR	1	None	0
2	2	AuthorID	INTEGER	0	None	0
3	3	Category	VARCHAR	1	None	0
4	4	PublishedYear	DATE	0	None	0
5	5	Price	DECIMAL	0	None	0

1.3 Create Customers Table

Table 3 : Customers

Col_Name	Type	Description
CustomerID	INT	to store the id of the customer PRIMARY KEY
CustomerName	VARCHAR	to store the name of the customer
Email	VARCHAR	to store the mail of the customer
Phone	VARCHAR	to store the phone number of the customer

We will Create the Table using the following command

```
CREATE TABLE IF NOT EXISTS Customers(
    CustomerID INTEGER PRIMARY KEY AUTOINCREMENT,
    CustomerName VARCHAR NOT NULL,
    Email VARCHAR UNIQUE NOT NULL,
    Phone VARCHAR,
);
```

Customers	
PK	CustomerID INT PRIMARY KEY AUTO_INCREMENT
	CustomerName VARCHAR(50) Email (VARCHAR) Phone (VARCHAR) Country (VARCHAR)

```
1 # Create Customers Table
2 cursor.execute('''
3 CREATE TABLE IF NOT EXISTS Customers(
4     CustomerID INTEGER PRIMARY KEY AUTOINCREMENT,
5     CustomerName TEXT NOT NULL,
6     Email TEXT UNIQUE NOT NULL,
7     Phone TEXT
8 );
9 ''')
10 conn.commit()
11
12 print("Customers table created successfully!")
13
14 # Display Customers table structure
15 customers_df = pd.read_sql_query("PRAGMA table_info(Customers);", conn)
16 customers_df
17
```

Customers table created successfully!

	cid	name	type	notnull	dflt_value	pk
0	0	CustomerID	INTEGER	0	None	1
1	1	CustomerName	TEXT	1	None	0
2	2	Email	TEXT	1	None	0
3	3	Phone	TEXT	0	None	0

1.4 Create Sales Table

Table 4 : Sales

Col_Name	Type	Description
SaleID	INT	to store the id of the sales PRIMARY KEY
BookID	INT	to link between it and the book that has been sold FOREIGN KEY
CustomerID	INT	to link between it and the customer who bought this book FOREIGN KEY
SaleDate	DATE	to store the date of the book when it was sold
Quantity	INT	to store the number of books have been sold

We will Create the Table using the following command

```
CREATE TABLE IF NOT EXISTS Sales(
    SaleID INTEGER PRIMARY KEY AUTOINCREMENT,
    BookID INTEGER,
    CustomerID INTEGER,
    SaleDate DATE,
    Quantity INTEGER NOT NULL DEFAULT 1,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Sales	
PK	SalesID INT PRIMARY KEY AUTO_INCREMENT
FK1	BookID INT CustomerID INT
	SaleDate DATE Quantity INT NOT NULL DEFAULT 1 FOREIGN KEY (BookID) REFERENCES Books(BookID)

```
[ ] 1 # Create Sales Table
2 cursor.execute('''
3 CREATE TABLE IF NOT EXISTS Sales(
4     SaleID INTEGER PRIMARY KEY AUTOINCREMENT,
5     BookID INTEGER,
6     CustomerID INTEGER,
7     SaleDate DATE,
8     Quantity INTEGER NOT NULL DEFAULT 1,
9     FOREIGN KEY (BookID) REFERENCES Books(BookID),
10    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
11 );
12 ''')
13 conn.commit()
14
15 print("Sales table created successfully!")
16
17 # Display Sales table structure
18 sales_df = pd.read_sql_query("PRAGMA table_info(Sales);", conn)
19 sales_df
```

→ Sales table created successfully!

	cid	name	type	notnull	dflt_value	pk
0	0	SaleID	INTEGER	0	None	1
1	1	BookID	INTEGER	0	None	0
2	2	CustomerID	INTEGER	0	None	0
3	3	SaleDate	DATE	0	None	0
4	4	Quantity	INTEGER	1	1	0

1.5 Create Inventory Table

Table 5 : Inventory

Col_Name	Type	Description
InventoryID	INT	to store the id of the inventory PRIMARY KEY
BookID	INT	to link between the inventory and the book FOREIGN KEY
StockQuantity	INT	to store quantity in the stock

We will Create the Table using the following command

```
CREATE TABLE IF NOT EXISTS Inventory(
    InventoryID INTEGER PRIMARY KEY AUTOINCREMENT,
    BookID INTEGER,
    StockQuantity INTEGER NOT NULL DEFAULT 0,
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

```
1 # Create Inventory Table
2 cursor.execute('''
3 CREATE TABLE IF NOT EXISTS Inventory(
4     InventoryID INTEGER PRIMARY KEY AUTOINCREMENT,
5     BookID INTEGER,
6     StockQuantity INTEGER NOT NULL DEFAULT 0,
7     FOREIGN KEY (BookID) REFERENCES Books(BookID)
8 );
9 ''')
10 conn.commit()
11
12 print("Inventory table created successfully!")
13
14 # Display Inventory table structure
15 inventory_df = pd.read_sql_query("PRAGMA table_info(Inventory);", conn)
16 inventory_df
17
```

Inventory table created successfully!

	cid	name	type	notnull	dflt_value	pk
0	0	InventoryID	INTEGER	0	None	1
1	1	BookID	INTEGER	0	None	0
2	2	StockQuantity	INTEGER	1	0	0

Inventory	
* PK	InventoryID INT PRIMARY KEY AUTO_INCREMENT
FK1	BookID INT
	StockQuantity INT FOREIGN KEY (BookID) REFERENCES Books(BookID)

Requirement Two

Insert Sample Data

2.1 Insert Data on Author Table

```
1 # Insert sample data into Authos
2 authors_data = [
3     ('Stephen King','USA'),
4     ('John Ronald Reuel Tolkien','England'),
5     ('William Shakespeare','England'),
6     ('Lewis Carroll','England'),
7     ('Agatha Christie','England'),
8     ('Arthur Conan Doyle','Scotland'),
9     ('George Orwell','UK'),
10    ('J.K. Rowling','UK'),
11    ('Ernest Hemingway','USA'),
12    ('Jane Austen','UK'),
13    ('Mark Twain','USA')
14 ]
15
16 # Insert the data from authors_data into the table Authors
17 cursor.executemany('''
18 INSERT INTO Authors (AuthorName, Country)
19 VALUES (?, ?);
20 ''', authors_data)
21 conn.commit()
22
23 print("Sample data inserted into Authors table successfully!")
24
25 # Display Authors table
26 authors_df = pd.read_sql_query("SELECT * FROM Authors;", conn)
27 authors_df.to_excel("authors.xlsx", index=False)
28 authors_df
```

⇒ Sample data inserted into Authors table successfully!

	AuthorID	AuthorName	Country
0	1	Stephen King	USA
1	2	John Ronald Reuel Tolkien	England
2	3	William Shakespeare	England
3	4	Lewis Carroll	England
4	5	Agatha Christie	England
5	6	Arthur Conan Doyle	Scotland
6	7	George Orwell	UK
7	8	J.K. Rowling	UK
8	9	Ernest Hemingway	USA
9	10	Jane Austen	UK
10	11	Mark Twain	USA

2.2 Insert Data on Books Table

```
[ ] 1 # Insert sample data into Books
2 books_data = [
3     ('Doctor Sleep', 1, 'Horror', '2013-12-1', 12.5),
4     ('Thinner ', 1, 'Drama', '1984-12-3', 13.5),
5     ('The Gunslinger', 1, 'Comedy', '1970-1-1', 11.5),
6     ('The Lord of the Rings', 2, 'Fantasy', '1954-2-2', 10.5),
7     ('The Hobbit', 2, 'Fantasy', '1937-1-2', 14.5),
8     ('The Fall of Númenor', 2, 'Fantasy', '2022-1-1', 12.5),
9     ('Hamlet', 3, 'Tragedy', '1599-1-1', 10.5),
10    ('The Tempest', 3, 'Drama', '1610-3-3', 14.5),
11    ('The Tragedie of Macbeth', 3, 'Tragedy', '1606-12-12', 11.5),
12    ('Alices Adventures in Wonderland', 4, 'Fantasy', '1903-12-1', 10.5),
13    ('Murder on the Orient Express', 5, 'Crime', '2008-1-1', 2.5),
14    ('The Adventures of Sherlock Holmes', 6, 'Detective', '1892-2-2', 11.5),
15    ('Animal Farm', 7, 'Novella', '1945-2-2', 10.5),
16    ('Harry Potter and the Deathly Hallows', 8, 'Fantasy', '2007-12-12', 9.5),
17    ('The Old Man and the Sea', 9, 'Novella', '1950-1-1', 12.5),
18    ('A Farewell to Arms', 9, 'War story', '1929-2-2', 11.5),
19    ('Pride and Prejudice', 10, 'Romance', '1813-1-1', 8.5),
20    ('Emma', 4, 'Romance', '1815-2-2', 7.99),
21    ('Life on the Mississippi', 5, 'Non-fiction', '1883-12-12', 11.99)
22 ]
23
24 cursor.executemany('''
25 INSERT INTO Books (Title, AuthorID, Category, PublishedYear, Price)
26 VALUES (?, ?, ?, ?, ?);
27 ''', books_data)
28 conn.commit()
29
30 print("Sample data inserted into Books table successfully!")
31
32 # Display Books table
33 books_df = pd.read_sql_query("SELECT * FROM Books;", conn)
34 books_df.to_excel("books.xlsx", index=False)
35 books_df
```

Sample data inserted into Books table successfully!

	BookID	Title	AuthorID	Category	PublishedYear	Price
0	1	Doctor Sleep	1	Horror	2013-12-1	12.50
1	2	Thinner	1	Drama	1984-12-3	13.50
2	3	The Gunslinger	1	Comedy	1970-1-1	11.50
3	4	The Lord of the Rings	2	Fantasy	1954-2-2	10.50
4	5	The Hobbit	2	Fantasy	1937-1-2	14.50
5	6	The Fall of Númenor	2	Fantasy	2022-1-1	12.50
6	7	Hamlet	3	Tragedy	1599-1-1	10.50
7	8	The Tempest	3	Drama	1610-3-3	14.50
8	9	The Tragedie of Macbeth	3	Tragedy	1606-12-12	11.50
9	10	Alices Adventures in Wonderland	4	Fantasy	1903-12-1	10.50
10	11	Murder on the Orient Express	5	Crime	2008-1-1	2.50
11	12	The Adventures of Sherlock Holmes	6	Detective	1892-2-2	11.50
12	13	Animal Farm	7	Novella	1945-2-2	10.50
13	14	Harry Potter and the Deathly Hallows	8	Fantnasy	2007-12-12	9.50
14	15	The Old Man and the Sea	9	Novella	1950-1-1	12.50
15	16	A Farewell to Arms	9	War story	1929-2-2	11.50
16	17	Pride and Prejudice	10	Romance	1813-1-1	8.50
17	18	Emma	4	Romance	1815-2-2	7.99
18	19	Life on the Mississippi	5	Non-fiction	1883-12-12	11.99

2.3 Insert Data on Inventory Table

```
1 # Insert sample data into Inventory
2 inventory_data = [
3     (1, 50),
4     (2, 30),
5     (3, 20),
6     (4, 15),
7     (5, 11),
8     (6, 13),
9     (7, 14),
10    (8, 15),
11    (9, 15),
12    (10, 10),
13    (11, 50),
14    (12, 30),
15    (13, 20),
16    (14, 15),
17    (15, 11),
18    (16, 13),
19    (17, 14),
20    (18, 15),
21    (19, 15)
22 ]
23
24 cursor.executemany('''
25 INSERT INTO Inventory (BookID, StockQuantity)
26 VALUES (?, ?);
27 ''', inventory_data)
28 conn.commit()
29
30 print("Sample data inserted into Inventory table successfully!")
31
32 # Display Inventory table
33 inventory_df = pd.read_sql_query("SELECT * FROM Inventory;", conn)
34 inventory_df.to_excel("inventory.xlsx", index=False)
35 inventory_df
```

➡ Sample data inserted into Inventory table successfully!


InventoryID	BookID	StockQuantity
0	1	1
1	2	2
2	3	3
3	4	4
4	5	5
5	6	6
6	7	7
7	8	8
8	9	9
9	10	10
10	11	11
11	12	12
12	13	13
13	14	14
14	15	15
15	16	16
16	17	17
17	18	18
18	19	19


2.4 Insert Data on Sales Table

We will Insert data in Customers Table using the following command

```
INSERT INTO Customers (CustomerName, Email, Phone) VALUES
('Basel Amr', 'baselamr52@gmail.com', '01000907482'),
('Omar Mohamed', 'Omar@gmail.com', '0987654321'),
('Abanoub', 'abanoub@gmail.com', '5678901234');
```

```
[ ] 1 # Insert sample data into Customers
2 customers_data = [
3     ('Basel Amr', 'baselamr52@gmail.com', '01000907482'),
4     ('Omar Mohamed', 'Omar@gmail.com', '0987654321'),
5     ('Abanoub', 'abanoub@gmail.com', '5678901234')
6 ]
7
8 cursor.executemany('''
9 INSERT INTO Customers (CustomerName, Email, Phone)
10 VALUES (?, ?, ?);
11 ''', customers_data)
12 conn.commit()
13
14 print("Sample data inserted into Customers table successfully!")
15
16 # Display Customers table
17 customers_df = pd.read_sql_query("SELECT * FROM Customers;", conn)
18 customers_df.to_excel("customers.xlsx", index=False)
19 customers_df
20
```

 Sample data inserted into Customers table successfully!

 Sample data inserted into Customers table successfully!

	CustomerID	CustomerName	Email	Phone
0	1	Basel Amr	baselamr52@gmail.com	01000907482
1	2	Omar Mohamed	Omar@gmail.com	0987654321
2	3	Abanoub	abanoub@gmail.com	5678901234

2.5 Insert Data on Inventory Table

```
1 # Insert sample data into Sales
2 sales_data = [
3     (1, 2, '2024-12-01', 5),
4     (2, 1, '2024-12-02', 3),
5     (3, 3, '2024-12-03', 7),
6     (4, 1, '2024-12-04', 4),
7     (5, 1, '2024-12-05', 6),
8     (6, 1, '2024-12-06', 2),
9     (7, 1, '2024-12-07', 5),
10    (8, 2, '2024-12-08', 3),
11    (9, 3, '2024-12-09', 4),
12    (10, 3, '2024-12-10', 1),
13    (11, 2, '2024-12-11', 2),
14    (12, 1, '2024-12-12', 3),
15    (13, 2, '2024-12-13', 4),
16    (14, 2, '2024-12-14', 5),
17    (15, 1, '2024-12-15', 6),
18    (16, 1, '2024-12-16', 1),
19    (17, 1, '2024-12-17', 1),
20    (18, 1, '2024-12-18', 3),
21    (19, 3, '2024-12-19', 4),
22    (3, 2, '2024-12-13', 7)
23 ]
24
25 cursor.executemany('''
26 INSERT INTO Sales (BookID, CustomerID, SaleDate, Quantity)
27 VALUES (?, ?, ?, ?);
28 ''', sales_data)
29 conn.commit()
30
31 print("Sample data inserted into Sales table successfully!")
32
33 # Display Sales table
34 sales_df = pd.read_sql_query("SELECT * FROM Sales;", conn)
35 sales_df.to_excel("sales.xlsx", index=False)
36 sales_df
```

Sample data inserted into Sales table successfully!

	SaleID	BookID	CustomerID	SaleDate	Quantity
0	1	1	2	2024-12-01	5
1	2	2	1	2024-12-02	3
2	3	3	3	2024-12-03	7
3	4	4	1	2024-12-04	4
4	5	5	1	2024-12-05	6
5	6	6	1	2024-12-06	2
6	7	7	1	2024-12-07	5
7	8	8	2	2024-12-08	3
8	9	9	3	2024-12-09	4
9	10	10	3	2024-12-10	1
10	11	11	2	2024-12-11	2
11	12	12	1	2024-12-12	3
12	13	13	2	2024-12-13	4
13	14	14	2	2024-12-14	5
14	15	15	1	2024-12-15	6
15	16	16	1	2024-12-16	1
16	17	17	1	2024-12-17	1
17	18	18	1	2024-12-18	3
18	19	19	3	2024-12-19	4
19	20	3	2	2024-12-13	7

Requirement Three

Query and Use SQL Joins

3.1 Write queries to retrieve all books and their authors

```
[ ] 1 # Query to retrieve all books and their authors
    2 query = '''
    3 SELECT
    4     Books.Title AS BookTitle,
    5     Authors.AuthorName AS Author
    6 FROM Books
    7 JOIN Authors ON Books.AuthorID = Authors.AuthorID;
    8 '''
    9
   10 books_authors_df = pd.read_sql_query(query, conn)
   11 books_authors_df.to_excel("books_authors.xlsx", index=False)
   12
   13 print("📖 All Books and Their Authors:")
   14 books_authors_df
   15
```



📖 All Books and Their Authors:

	BookTitle	Author
0	Doctor Sleep	Stephen King
1	Thinner	Stephen King
2	The Gunslinger	Stephen King
3	The Lord of the Rings	John Ronald Reuel Tolkien
4	The Hobbit	John Ronald Reuel Tolkien
5	The Fall of Númenor	John Ronald Reuel Tolkien
6	Hamlet	William Shakespeare
7	The Tempest	William Shakespeare
8	The Tragedie of Macbeth	William Shakespeare
9	Alices Adventures in Wonderland	Lewis Carroll
10	Murder on the Orient Express	Agatha Christie
11	The Adventures of Sherlock Holmes	Arthur Conan Doyle
12	Animal Farm	George Orwell
13	Harry Potter and the Deathly Hallows	J.K. Rowling
14	The Old Man and the Sea	Ernest Hemingway
15	A Farewell to Arms	Ernest Hemingway
16	Pride and Prejudice	Jane Austen
17	Emma	Lewis Carroll
18	Life on the Mississippi	Agatha Christie

3.2 Write a query to find total sales per book

✓ 3.2. Write a query to find total sales per book.

```
[ ] 1 # Total sales per book
    2 query = '''
    3 SELECT
    4     Books.Title AS BookTitle,
    5     SUM(Sales.Quantity) AS TotalSold,
    6     SUM(Sales.Quantity * Books.Price) AS TotalRevenue
    7 FROM Sales
    8 JOIN Books ON Sales.BookID = Books.BookID
    9 GROUP BY Books.Title
   10 ORDER BY TotalRevenue DESC;
   11 '''
   12 total_sales_df = pd.read_sql_query(query, conn)
   13 total_sales_df.to_excel("total_sales.xlsx", index=False)
   14
   15 print("💰 Total Sales per Book:")
   16 total_sales_df
```

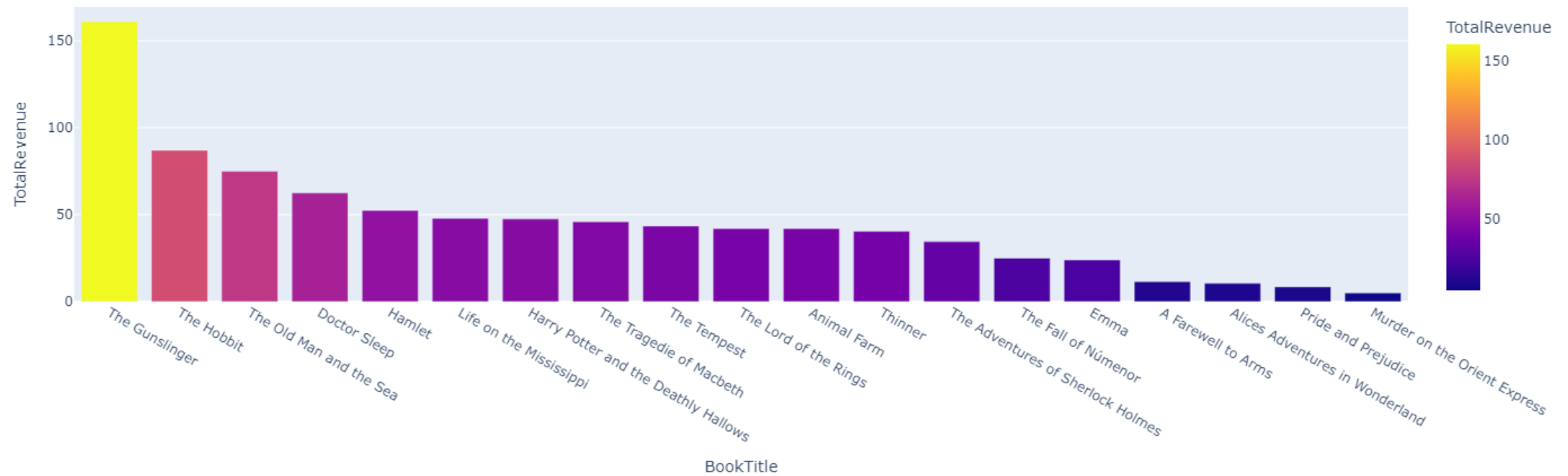
➡ 💰 Total Sales per Book:

	BookTitle	TotalSold	TotalRevenue
0	The Gunslinger	14	161.00
1	The Hobbit	6	87.00
2	The Old Man and the Sea	6	75.00
3	Doctor Sleep	5	62.50
4	Hamlet	5	52.50
5	Life on the Mississippi	4	47.96
6	Harry Potter and the Deathly Hallows	5	47.50
7	The Tragedie of Macbeth	4	46.00
8	The Tempest	3	43.50
9	The Lord of the Rings	4	42.00
10	Animal Farm	4	42.00
11	Thinner	3	40.50
12	The Adventures of Sherlock Holmes	3	34.50
13	The Fall of Númenor	2	25.00
14	Emma	3	23.97
15	A Farewell to Arms	1	11.50
16	Alices Adventures in Wonderland	1	10.50
17	Pride and Prejudice	1	8.50
18	Murder on the Orient Express	2	5.00

Sprints

3.2 Visualization

💰 Top-Selling Books by Revenue



3.3 Write a query using join to combine data from books and Sales

```
[17] 1 # Join Books and Sales
      2 query = '''
      3 SELECT
      4     Sales.SaleID, Books.Title AS BookTitle,
      5     Sales.Quantity, Sales.SaleDate,
      6     (Sales.Quantity * Books.Price) AS SaleAmount
      7 FROM Sales
      8 JOIN Books ON Sales.BookID = Books.BookID;
      9 '''
     10 sales_summary_df = pd.read_sql_query(query, conn)
     11 sales_summary_df.to_excel("sales_summary.xlsx", index=False)
     12
     13 print("📊 Sales Summary:")
     14 sales_summary_df
```

[17] 📊 Sales Summary:



	SaleID	BookTitle	Quantity	SaleDate	SaleAmount
0	1	Doctor Sleep	5	2024-12-01	62.50
1	2	Thinner	3	2024-12-02	40.50
2	3	The Gunslinger	7	2024-12-03	80.50
3	4	The Lord of the Rings	4	2024-12-04	42.00
4	5	The Hobbit	6	2024-12-05	87.00
5	6	The Fall of Númenor	2	2024-12-06	25.00
6	7	Hamlet	5	2024-12-07	52.50
7	8	The Tempest	3	2024-12-08	43.50
8	9	The Tragedie of Macbeth	4	2024-12-09	46.00
9	10	Alices Adventures in Wonderland	1	2024-12-10	10.50
10	11	Murder on the Orient Express	2	2024-12-11	5.00
11	12	The Adventures of Sherlock Holmes	3	2024-12-12	34.50
12	13	Animal Farm	4	2024-12-13	42.00
13	14	Harry Potter and the Deathly Hallows	5	2024-12-14	47.50
14	15	The Old Man and the Sea	6	2024-12-15	75.00
15	16	A Farewell to Arms	1	2024-12-16	11.50
16	17	Pride and Prejudice	1	2024-12-17	8.50
17	18	Emma	3	2024-12-18	23.97
18	19	Life on the Mississippi	4	2024-12-19	47.96
19	20	The Gunslinger	7	2024-12-13	80.50



3.3 Visualization

Sales Trend Over Time



3.4 Top 5 Authors by books

✓ 3.4.Top 5 Authors by Book Count

```
[29] 1 query = '''
      2 SELECT
      3     Authors.AuthorName,
      4     COUNT(Books.BookID) AS BookCount
      5 FROM Authors
      6 JOIN Books ON Authors.AuthorID = Books.AuthorID
      7 GROUP BY Authors.AuthorName
      8 ORDER BY BookCount DESC
      9 LIMIT 5;
     10 '''
     11 top_authors_df = pd.read_sql_query(query, conn)
     12 display(top_authors_df)
     13
```

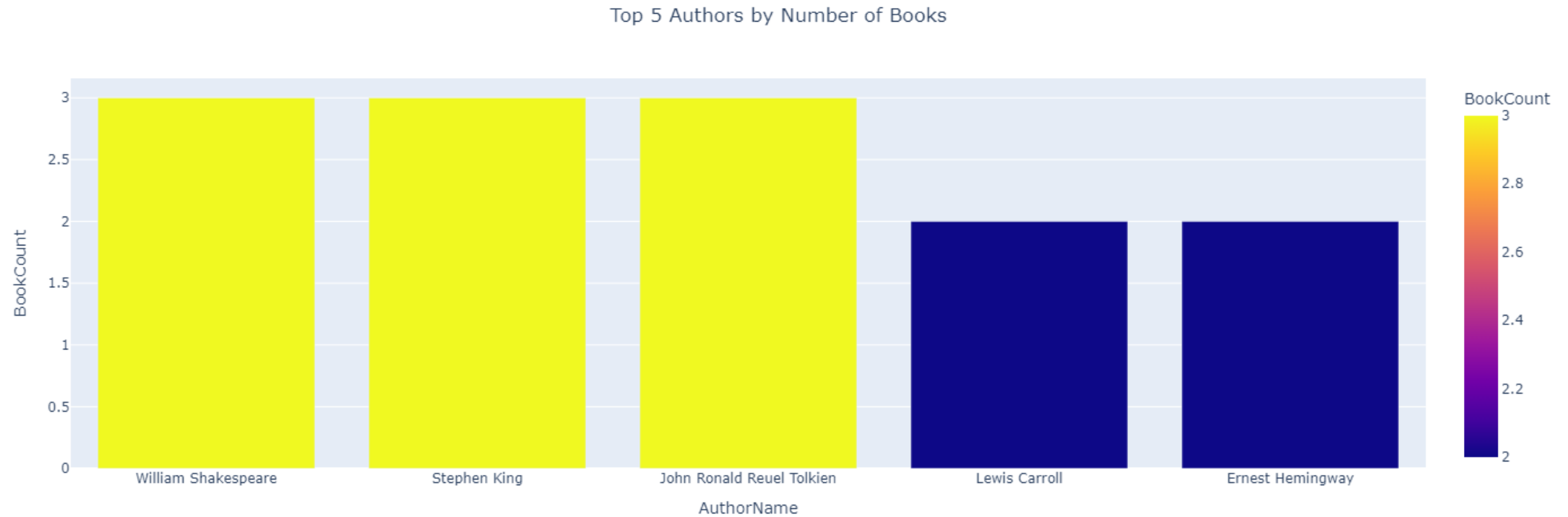


	AuthorName	BookCount
0	William Shakespeare	3
1	Stephen King	3
2	John Ronald Reuel Tolkien	3
3	Lewis Carroll	2
4	Ernest Hemingway	2



Sprints

3.5 Visualization





3.5 Inventory Overview

✓ 3.5.Inventory Overview

```
[20] 1 query = '''
      2 SELECT
      3     Books.Title AS BookTitle,
      4     Inventory.StockQuantity AS InitialStock,
      5     IFNULL(SUM(Sales.Quantity), 0) AS TotalSold,
      6     (Inventory.StockQuantity - IFNULL(SUM(Sales.Quantity), 0)) AS StockLeft
      7 FROM Inventory
      8 JOIN Books ON Inventory.BookID = Books.BookID
      9 LEFT JOIN Sales ON Inventory.BookID = Sales.BookID
     10 GROUP BY Books.Title, Inventory.StockQuantity;
     11 '''
     12 inventory_df = pd.read_sql_query(query, conn)
     13 inventory_df.to_excel("inventory_status.xlsx", index=False)
     14 print("📦 Inventory Status:")
     15 inventory_df
```

[20] 📦 Inventory Status:

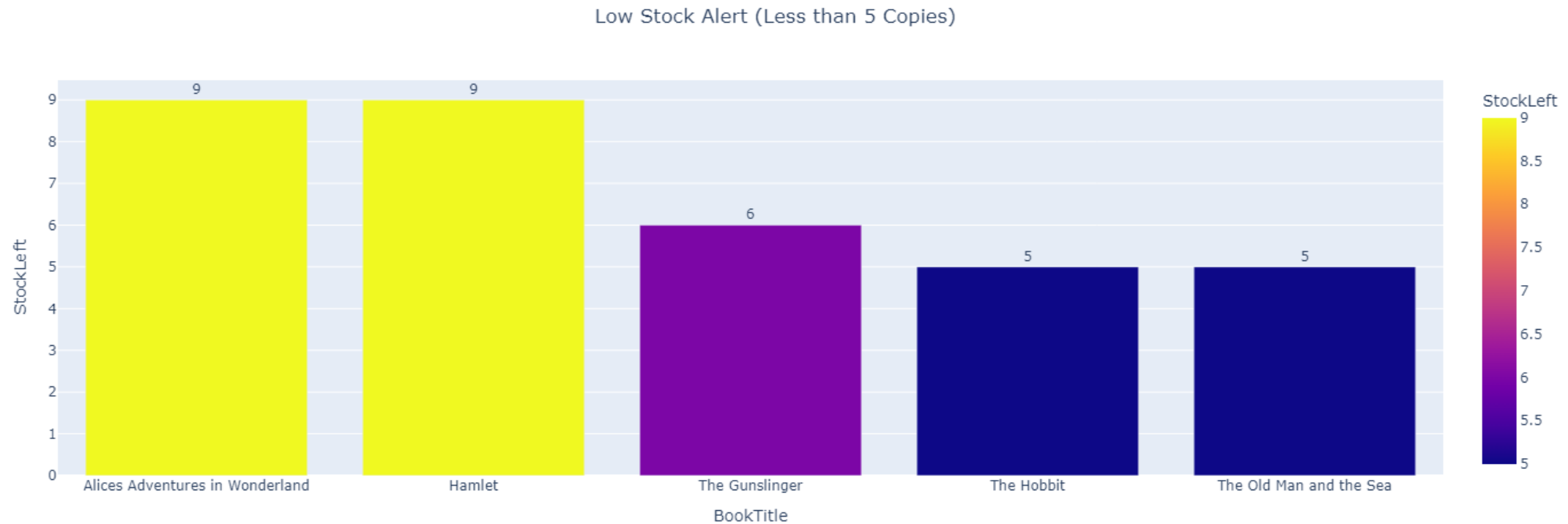
	BookTitle	InitialStock	TotalSold	StockLeft
0	A Farewell to Arms	13	1	12
1	Alices Adventures in Wonderland	10	1	9
2	Animal Farm	20	4	16
3	Doctor Sleep	50	5	45
4	Emma	15	3	12
5	Hamlet	14	5	9
6	Harry Potter and the Deathly Hallows	15	5	10
7	Life on the Mississippi	15	4	11
8	Murder on the Orient Express	50	2	48
9	Pride and Prejudice	14	1	13
10	The Adventures of Sherlock Holmes	30	3	27
11	The Fall of Númenor	13	2	11
12	The Gunslinger	20	14	6
13	The Hobbit	11	6	5
14	The Lord of the Rings	15	4	11
15	The Old Man and the Sea	11	6	5
16	The Tempest	15	3	12
17	The Tragedie of Macbeth	15	4	11
18	Thinner	30	3	27



3.5 Low Stock Alert

```
1 # Low Stock Alert
2 low_stock_df = inventory_df[inventory_df['StockLeft'] < 10]
3 fig = px.bar(
4     low_stock_df,
5     x='BookTitle',
6     y='StockLeft',
7     title='Low Stock Alert (Less than 5 Copies)',
8     color='StockLeft',
9     text='StockLeft'
10 )
11 fig.update_traces(texttemplate='%{text}', textposition='outside')
12 fig.show()
```

3.5 Visualization



3.6 Top 5 Customers by Purchase Amount

✓ 3.6 Top Customers by Purchase Amount

```
[23] 1 # Top Customers by Purchase Amount
      2 query = '''
      3 SELECT
      4     Customers.CustomerName,
      5     IFNULL(SUM(Sales.Quantity * Books.Price), 0) AS TotalSpent
      6 FROM Customers
      7 JOIN Sales ON Customers.CustomerID = Sales.CustomerID
      8 JOIN Books ON Sales.BookID = Books.BookID
      9 GROUP BY Customers.CustomerID, Customers.CustomerName
     10 ORDER BY TotalSpent DESC
     11 LIMIT 10;
     12 '''
     13
     14 top_customers_df = pd.read_sql_query(query, conn)
     15 top_customers_df.to_excel("top_customers.xlsx", index=False)
     16
     17 print("🏆 Top Customers by Purchase Amount:")
     18 top_customers_df
     19
```

🔄 🏆 Top Customers by Purchase Amount:

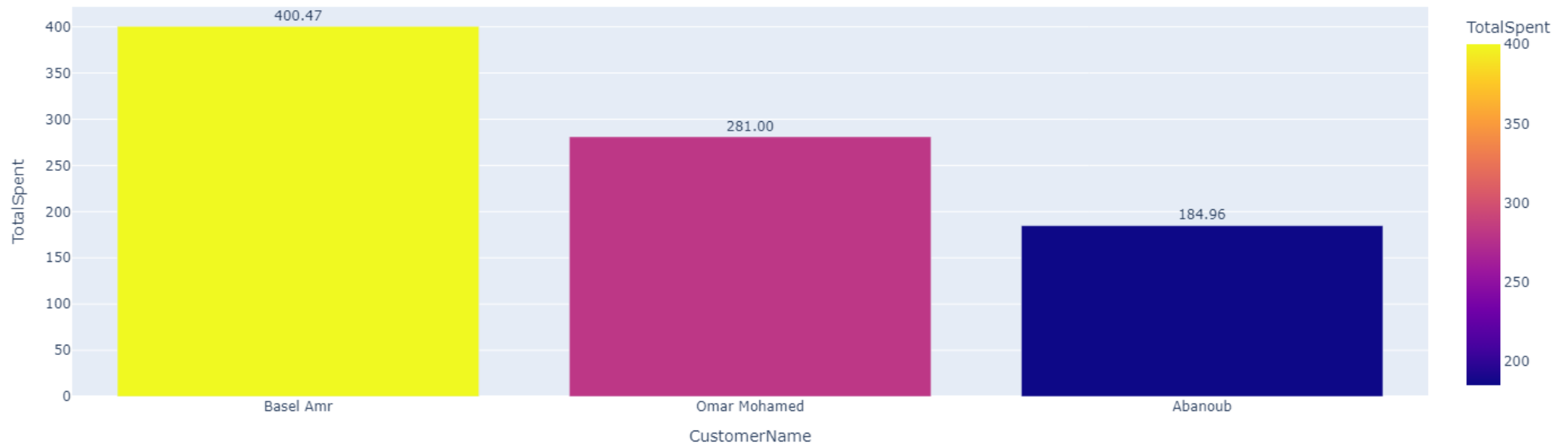
	CustomerName	TotalSpent
0	Basel Amr	400.47
1	Omar Mohamed	281.00
2	Abanoub	184.96





3.6 Visualization

🏆 Top Customers by Purchase Amount



3.7 Bestsellers based on categories

✓ 3.7 Bestseller Categories

```
[25] 1 # Bestseller Categories
      2 query = '''
      3 SELECT
      4     Books.Category,
      5     IFNULL(SUM(Sales.Quantity), 0) AS TotalSold
      6 FROM Books
      7 LEFT JOIN Sales ON Books.BookID = Sales.BookID
      8 GROUP BY Books.Category
      9 ORDER BY TotalSold DESC;
     10 '''
     11
     12 bestseller_categories_df = pd.read_sql_query(query, conn)
     13 bestseller_categories_df.to_excel("bestseller_categories.xlsx", index=False)
     14
     15 print("Bestseller Categories:")
     16 bestseller_categories_df
```

⇌ Bestseller Categories:

	Category	TotalSold
0	Comedy	14
1	Fantasy	13
2	Novella	10
3	Tragedy	9
4	Drama	6
5	Horror	5
6	Fantrasy	5
7	Romance	4
8	Non-fiction	4
9	Detective	3
10	Crime	2
11	War story	1

3.7 Visualization

Bestseller Categories

