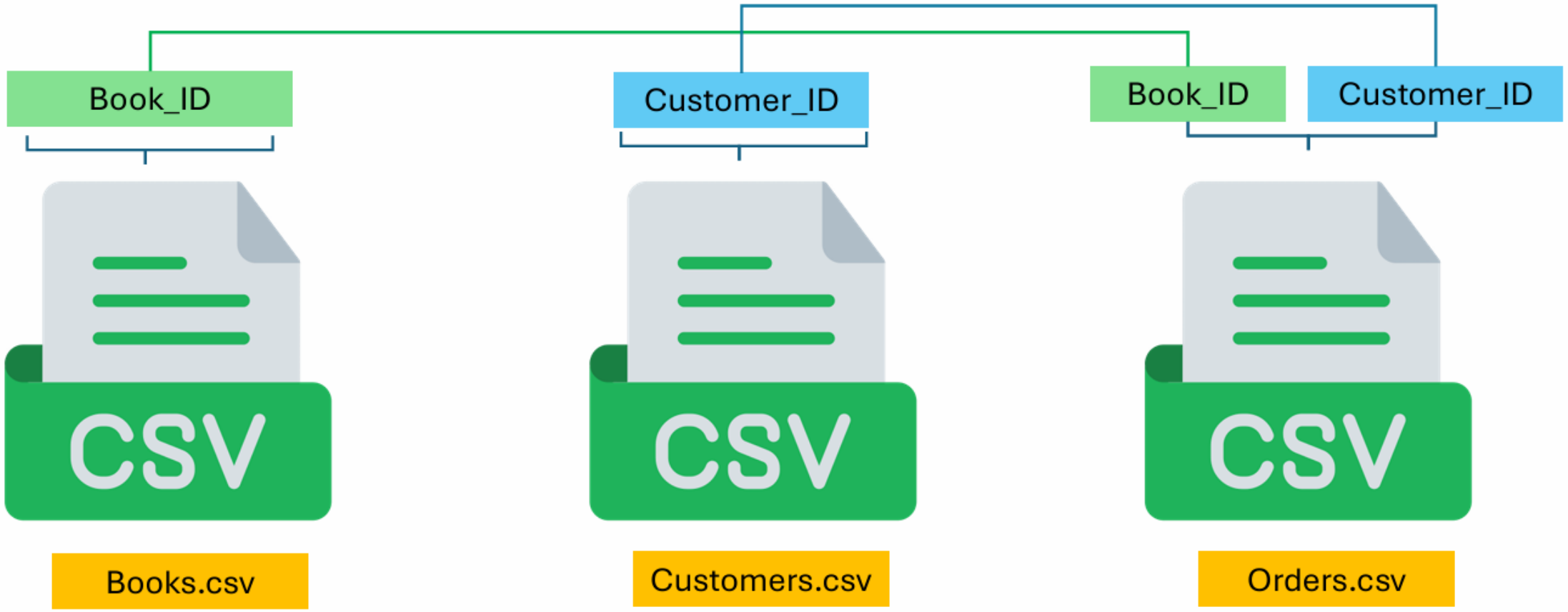
SQL Project on Online Book store

3 CSV Files

Tables must have at least one common column with same column name and same data type



Basic Queries

1) Retrieve all books in the "Fiction" genre

2) Find books published after the year 1950

3) List all customers from the Canada

4) Show orders placed in November 2023

5) Retrieve the total stock of books available

6) Find the details of the most expensive book

7) Show all customers who ordered more than 1 quantity of a book

8) Retrieve all orders where the total amount exceeds $20

9) List all genres available in the Books table

10) Find the book with the lowest stock

11) Calculate the total revenue generated from all orders

Advance Queries

1) Retrieve the total number of books sold for each genre
2) Find the average price of books in the "Fantasy" genre
3) List customers who have placed at least 2 orders
4) Find the most frequently ordered book
5) Show the top 3 most expensive books of 'Fantasy' Genre
6) Retrieve the total quantity of books sold by each author
7) List the cities where customers who spent over $30 are located
8) Find the customer who spent the most on orders
9) Calculate the stock remaining after fulfilling all orders

```sql
-- Basic Queries:

-- 1) Retrieve all books in the "Fiction" genre:
SELECT *
FROM books
WHERE genre = 'Fiction';

-- 2) Find books published after the year 1950:
SELECT *
FROM books
WHERE published_year > '1950';

-- 3) List all customers from the Canada:
SELECT *
FROM customers
WHERE country = 'Canada';

-- 4) Show orders placed in November 2023:
SELECT *
FROM Orders
WHERE Order_date BETWEEN '2023-11-01'
AND '2023-11-30';
```

```sql
-- 5) Retrieve the total stock of books available:
SELECT
SUM(stock) AS total_stock
FROM Books;

-- 6) Find the details of the most expensive book:
SELECT *
FROM Books
ORDER BY price DESC
LIMIT 1;

-- 7) Show all customers who ordered more than 1 quantity of a book:
SELECT *
FROM orders
WHERE quantity > 1;

-- 8) Retrieve all orders where the total amount exceeds $20:
SELECT *
FROM orders
WHERE total_amount > 20;
```

```sql
-- 9) List all genres available in the Books table:
SELECT
DISTINCT genre
FROM books;

-- 10) Find the book with the lowest stock:
SELECT *
FROM books
ORDER BY stock ASC
LIMIT 1;

-- 11) Calculate the total revenue generated from all orders:
SELECT
SUM(total_amount) AS revenue
FROM orders;
```

```sql
-- Advance Queries :

-- 1) Retrieve the total number of books sold for each genre:

SELECT b.genre,
SUM(o.quantity) AS total_book
FROM orders O
join books b
ON o.book_id = b.book_id
GROUP BY b.genre;

-- 2) Find the average price of books in the "Fantasy" genre:

SELECT
AVG(price) AS average_price
FROM books
WHERE genre = 'Fantasy'
GROUP BY genre;
```

```sql
-- 3) List customers who have placed at least 2 orders:
SELECT o.customer_id, c.name,
COUNT(o.order_id) AS order_count
FROM orders o
JOIN customers c
ON o.customer_id  = c.customer_id
GROUP BY o.customer_id, c.name
HAVING COUNT(order_id) >= 2;

4) Find the most frequently ordered book:

SELECT o.book_id, b.title,
COUNT(o.order_id) AS order_count
FROM orderS o
JOIN books b
ON o.book_id = b.book_id
GROUP BY o.book_id, b.title
ORDER BY order_count DESC
LIMIT 1;
```

```sql
-- 5) Show the top 3 most expensive books of 'Fantasy' Genre:
SELECT *
FROM books
WHERE  genre = 'Fantasy'
ORDER BY price DESC
LIMIT 3;

6) Retrieve the total quantity of books sold by each author:

SELECT b.author,
SUM(o.quantity) AS total_book_sold
FROM orders o
JOIN books b
ON b.book_id = o.book_id
GROUP BY b.author;

-- 7) List the cities where customers who spent over $30 are located:
SELECT
DISTINCT c.city, total_amount
FROM orders o
JOIN customers c
ON o.customer_id = c.customer_id
WHERE o.totaL_amount > 30;
```

8) Find the customer who spent the most on orders:

```
SELECT c.customer_id, c.name,
SUM(o.total_amount) AS total_spent
FROM orders o
JOIN customers c
ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC
LIMIT 1;

--9) Calculate the stock remaining after fulfilling all orders:

SELECT b.book_id, b.title, b.stock,
COALESCE (SUM(o.quantity),0) AS order_quantity, b.stock - COALESCE (sum(o.quantity),0) AS
remaining_quantity
FROM books b
LEFT JOIN orders o
ON b.book_id = o.book_id
GROUP BY b.book_id
ORDER BY b.book_id;
```