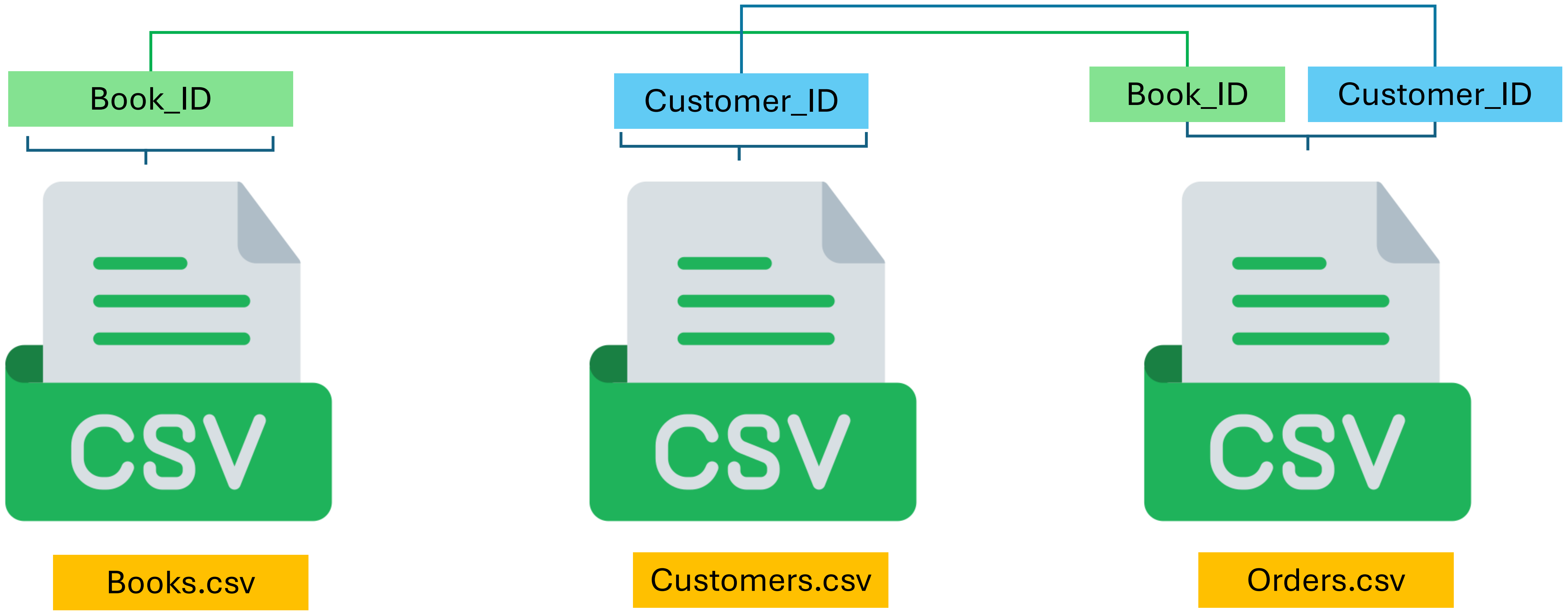
SQL Project on Online Book store

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 Project : Online Book Store

Queries

```
CREATE DATABASE OnlineBookStore;

-- Switch to the database

\c OnlineBookStore;

DROP TABLE IF EXISTS books;

CREATE TABLE books(

            Book_ID SERIAL PRIMARY KEY,

   Title VARCHAR(100),

   Author VARCHAR(100),

   Genre VARCHAR(50),

   Published_Year INT,

   Price NUMERIC(10, 2),

   Stock INT

);


COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)

FROM 'C:\Users\rushi\Desktop\SQL Practice\Books.csv'

DELIMITER ','

CSV HEADER;


CREATE TABLE customers(

Customer_ID SERIAL PRIMARY KEY,

Name  VARCHAR(100),

Email   VARCHAR(100),

Phone VARCHAR(15),
City      VARCHAR(50),
Country           VARCHAR(100)
);
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'C:\Users\rushi\Desktop\SQL Practice\Customers.csv'
DELIMITER ','
CSV HEADER;

CREATE TABLE orders(
Order_ID SERIAL PRIMARY KEY,
Customer_ID INT REFERENCES Customers(Customer_ID),
Book_ID INT REFERENCES Books(Book_ID),
Order_Date DATE,
Quantity INT,
Total_Amount NUMERIC(10,2)
);

COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM 'C:\Users\rushi\Desktop\SQL Practice\Orders.csv'
DELIMITER ','
CSV HEADER

SELECT * FROM books;
SELECT * FROM customers;
SELECT * FROM orders;
```

-- 1) Retrieve all books in the "Fiction" genre:

```
SELECT book_id, title, genre

FROM books

WHERE genre = 'Fiction';
```

-- 2) Find books published after the year 1950:

```
SELECT book_id, title, genre, published_year

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 >= '2023-11-01' AND order_date <= '2023-11-30';

--Or

SELECT * FROM orders

WHERE order_date BETWEEN '2023-11-01' AND '2023-11-30';
```

-- 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
```

```sql
ORDER BY price DESC LIMIT 1;
```

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

```sql
SELECT * FROM orders

WHERE quantity > 1;
```

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

```sql
SELECT * FROM orders

WHERE total_amount > 20;
```

-- 9) List all genres available in the Books table:

```sql
SELECT DISTINCT genre FROM books;

SELECT DISTINCT genre, title FROM books ORDER BY genre;
```

-- 10) Find the book with the lowest stock:

```sql
SELECT * FROM books ORDER BY stock LIMIT 1;
```

-- 11) Calculate the total revenue generated from all orders:

```sql
SELECT SUM(total_amount) AS total_revenue FROM orders;
```

-- Advance Questions :

```sql
SELECT * FROM books;

SELECT * FROM customers;

SELECT * FROM orders;
```

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

SELECT * FROM orders;


SELECT b. genre, SUM(o. quantity) AS total_books_sold

FROM orders o JOIN books b

ON o. book_id = b. book_id

GROUP BY b. genre;


--Q1.1 with revenue

SELECT b. genre, SUM(o. quantity) AS total_books_sold, SUM(o. total_amount) AS total_books_revenue

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 * FROM books;

SELECT AVG(price) AS Avg_price

FROM books

WHERE genre = 'Fantasy';


-- 3) List customers who have placed at least 2 orders:

SELECT customer_id, COUNT(order_id) AS order_count

FROM orders

GROUP BY customer_id

HAVING COUNT(order_id)>= 2;
```

--Q 3.1) with curtomer name

```
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(o.order_id)>= 2;
```

-- 4) Find the most frequently ordered book:

```
SELECT book_id, COUNT(order_id) AS order_count

FROM orders

GROUP BY book_id ORDER BY order_count  DESC LIMIT 1;
```

--Q 4.1) With book name

```
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;
```

-- 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_bookssold

FROM orders o JOIN books b ON o.book_id = b.book_id

GROUP BY b.author;
```

--For an author

```
SELECT SUM(o.quantity) AS total_books_sold

FROM orders o JOIN books b ON o.book_id = b.book_id

WHERE b.author = 'Tracy Parker';
```

-- 7) List the cities where customers who spent over $30 are located:

```
SELECT DISTINCT c.city, o.total_amount

FROM orders o JOIN customers c

ON o.customer_id = c.customer_id

WHERE o.total_amount > 30;


SELECT * FROM customers;

SELECT * FROM orders;
```

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

```
SELECT c.customer_id, c.name, SUM(o.total_amount) AS highest_spent

FROM orders o JOIN customers c ON c.customer_id = o.customer_id

GROUP BY c.customer_id, c.name

ORDER BY highest_spent DESC LIMIT 1;
```

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

```
SELECT b.book_id, b.title, b.stock, COALESCE(SUM (quantity),0) AS order_quantity

FROM books b LEFT JOIN orders o

ON b.book_id = o.book_id

GROUP BY b.book_id;
```

--ANS

```
SELECT b.book_id, b.title, b.stock, COALESCE(SUM (quantity),0) AS order_quantity,

b.stock - COALESCE(SUM (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;
```