PostgreSQL SQL Project - 20 Practice Queries

```
--Create tables

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
);

SELECT * FROM BOOKS ;

DROP TABLE IF EXISTS orders;

DROP TABLE IF EXISTS Customers;
CREATE TABLE Customers(
 Customer_ID INT PRIMARY KEY,
 Name VARCHAR(100),
 Email VARCHAR(100),
 Phone VARCHAR(15),
 City VARCHAR(50),
 Country VARCHAR(150)
);

SELECT * FROM customers;

DROP TABLE IF EXISTS orders;
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)
);

SELECT * FROM orders;

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

-- Import Data into Books Table
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM                                                                    'C:\Users\AHAD
SK\Downloads\SQL_Resume_Project-main\SQL_Resume_Project-main\Books.csv'
CSV HEADER;
```

```sql
-- Import Data into Customers Table
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'D:\Course Updates\30 Day Series\SQL\CSV\Customers.csv'
CSV HEADER;

-- Import Data into Orders Table
COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM 'D:\Course Updates\30 Day Series\SQL\CSV\Orders.csv'
CSV HEADER;


-- 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'
 ORDER BY published_year;

-- 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'
 ORDER BY order_date;


-- 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
   AS Availaible_genre
   FROM Books;

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

  SELECT * FROM books
  ORDER BY stock
  LIMIT 1;

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

 SELECT SUM(total_amount)
 AS Total_revenue
 FROM orders;

-- Advance Questions :

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


-- 2) Find the average price of books in the "Fantasy" genre:
 SELECT AVG(price) AS Average_Price
 FROM Books
 WHERE genre = 'Fantasy';

-- 3) List customers who have placed at least 2 orders:
SELECT * FROM customers;
SELECT * FROM ORDERS;

 SELECT o.customer_id ,c.name,COUNT(o.order_id) AS Total_order
 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 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;

 SELECT * FROM Books;

-- 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_books_Sold
 FROM books b
 JOIN
 orders o
 ON o.book_id=b.book_id
 GROUP BY b.author;




-- 7) List the cities where customers who spent over $30 are located:
SELECT * FROM customers;
SELECT * FROM orders;

SELECT DISTINCT c.city , o.total_amount
FROM customers c
JOIN
orders o
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 * FROM Books;
SELECT * FROM orders;

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

```
b.stock - COALESCE (SUM(o.quantity),0) AS Remaining_Quantity
FROM books b
JOIN orders o ON b.book_id = o.book_id
GROUP BY b.book_id ORDER BY b.book_id;
```

```
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;
```