@Aasif shaikh

ONLINE BOOKSTORE ANALYSIS USING SQL

PROJECT

OVERVIEW

This SQL project explores a small bookstore database containing customers, books, and orders. The goal is to practice and demonstrate SQL skills by analyzing sales, customer behavior, inventory, and order trends.
I worked with three connected tables — books, customers, and orders — and wrote a series of queries covering JOINs, filtering, aggregations, and calculations. Each question helps uncover meaningful insights such as best-selling books, total revenue, customer purchase patterns, and stock analysis.
This project is designed both as a learning exercise and a resource for others to try the same queries and build their SQL skills.

@Aasif shaikh

@Aasif shaikh

TABLE SCHEMA

customers (customer_id, name, email, phone, city, country)

|

orders (order_id, customer_id, book_id, order_date, quantity, total_amount)

|

books (book_id, title, author, genre, published_year, price, stock)

INTERMEDIATE LEVEL

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

```sql
SELECT
    book_id, title, author, genre
FROM
    books
WHERE
    genre = 'fiction';
```

2) Find books published after the year 1950:

```sql
SELECT
    title, published_year
FROM
    books
WHERE
    published_year > '1950';
```

3) List all customers from the Canada:

```sql
SELECT
    customer_id, name, country
FROM
    customers
WHERE
    country = 'canada';
```

4) Show orders placed in November 2023:

```sql
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:

```sql
SELECT
    ROUND(SUM(price * stock), 2) AS total_cost_of_book_avl
FROM
    books;
```

6) Find the details of the most expensive book:

```sql
SELECT
    *
FROM
    books
ORDER BY price DESC
LIMIT 1;
```

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

```sql
SELECT
    c.customer_id, c.name, o.quantity
FROM
    customers c
        JOIN
    orders o ON c.customer_id = o.customer_id
WHERE
    o.quantity > 5;
```

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

```sql
SELECT
    order_id, order_date, quantity, total_amount
FROM
    orders
WHERE
    total_amount > 20;
```

9) List all genres available in the Books table:

```
SELECT DISTINCT
    genre
FROM
    books;
```

10) Find the book with the lowest stock:

```sql
SELECT
    title, MIN(stock) AS lowest_stock
FROM
    books
GROUP BY title
ORDER BY lowest_stock;
```

11) Calculate the total revenue generated from all orders:

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

ADVANCE LEVEL

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

```sql
SELECT
    genre, SUM(quantity) AS book_sold
FROM
    books b
        JOIN
    orders o ON b.book_id = o.book_id
GROUP BY genre
ORDER BY book_sold DESC;
```

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

```sql
SELECT
    ROUND(AVG(price), 2) AS avg_price
FROM
    books
WHERE
    genre = 'fantasy';
```

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

```sql
SELECT
    *
FROM
    (SELECT
        c.name, COUNT(*) AS order_count
    FROM
        customers c
    JOIN orders o ON c.customer_id = o.customer_id
    GROUP BY c.name) AS oc
WHERE
    order_count >= 2
ORDER BY order_count DESC;
```

4) Find the most frequently ordered book:

```sql
SELECT
    b.title, b.Genre, COUNT(o.Book_ID) AS total_ordered
FROM
    books b
        JOIN
    orders o ON b.book_id = o.book_id
GROUP BY b.Title , b.genre
ORDER BY total_ordered DESC
LIMIT 10;
```

5) Show the top 3 most expensive books of 'Fantasy' Genre :

```sql
SELECT
    title, genre, price
FROM
    books
WHERE
    genre = 'fantasy'
ORDER BY price DESC
LIMIT 3;
```

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

```sql
SELECT
    b.author, SUM(o.Quantity) AS total_quantity_sold
FROM
    books b
        JOIN
    orders o ON b.book_id = o.book_id
GROUP BY b.author
ORDER BY total_quantity_sold DESC;
```

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

```sql
SELECT
    *
FROM
    (SELECT
        c.city, SUM(Total_Amount) AS total_sum
    FROM
        customers c
    JOIN orders o ON c.customer_id = o.customer_id
    GROUP BY c.City) AS a
WHERE
    total_sum > 50;
```

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

```sql
SELECT
    c.name, SUM(total_amount) AS total_spent
FROM
    customers c
        JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY c.name
ORDER BY total_spent DESC
LIMIT 10;
```

9) Calculate the stock remaining after fulfilling all orders:

```sql
SELECT
    b.title, b.author, (b.Stock - o.Quantity) AS remaining_stock
FROM
    books b
        JOIN
    orders o ON b.book_id = o.book_id
ORDER BY remaining_stock;
```

Found this helpful ?

Save & Follow for more content

github.com/Aasifshaikh-00000

Like Comment share