SQL Project: Online Bookstore

Name: HASNAIN MOAVIA **Instructor Name**: Satish Dhawale

Date: 25-Aug-2025

Overview

This project demonstrates the creation and management of an online bookstore database using SQL. The database consists of three CSV files: `Books.csv`, `Customers.csv`, and `Orders.csv`. Tables are related via `Book_ID` and `Customer_ID`.

Objective

- **Design a relational database for an online bookstore**
- **Perform CRUD operations and run queries to analyze data**
- **Generate insights like revenue, top-selling books, and customer activity**
- **Learn to manage CSV imports into SQL tables**

SQL Queries

1- Retrieve all books in the 'Fiction' genre:

SQL QUERY:

```sql
SELECT * FROM BOOKS
WHERE genre = 'Fiction';
```

RESULT:

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	
2	22	Multi-layered optimizing migration	Wesley Escobar	Fiction	1908	39.23	
3	28	Expanded analyzing portal	Lisa Coffey	Fiction	1941	37.51	
4	29	Quality-focused multi-tasking challenge	Katrina Underwood	Fiction	1905	31.12	1
5	31	Implemented encompassing conglomerati...	Melissa Taylor	Fiction	2010	21.23	
6	39	Optimized national process improvement	Megan Goodwin	Fiction	1978	10.99	
7	40	Adaptive didactic interface	Natalie Gonzalez	Fiction	1923	25.97	
8	47	Reverse-engineered directional conglomer...	John Christian	Fiction	2006	20.37	
9	62	Re-contextualized real-time strategy	Nicole Lynch	Fiction	1953	26.34	
10	63	Polarized heuristic database	Franklin Mack	Fiction	1989	22.38	
11	100	Synchronized client-server service-desk	James Alvarado	Fiction	1906	49.89	
12	116	Multi-tiered foreground contingency	Jamie Gates	Fiction	1938	41.82	
13	125	Public-key analyzing Graphic Interface	Abigail Madden	Fiction	1990	32.41	
14	130	Realigned context-sensitive pricing structure	Jason Rodriguez	Fiction	2004	6.64	
15	134	Polarized bandwidth-monitored throughput	Linda Newman	Fiction	1955	35.72	
16	142	Multi-tiered responsive parallelism	Amanda Wilson	Fiction	1940	48.96	
17	143	Networked multimedia support	Nancy Goodman	Fiction	2012	43.65	

2. Find books published after 1950:

SQL QUERY:

```
SELECT * FROM BOOKS
WHERE published_year > 1950;
```

RESULT:

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)
1	2	Persevering reciprocal knowledge user	Mario Moore	Fantasy	1971	35
2	4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43
3	5	Adaptive 5thgeneration encoding	Juan Miller	Fantasy	1956	10
4	6	Advanced encompassing implementation	Bryan Morgan	Biography	1985	6
5	8	Persistent local encoding	Troy Cox	Science Fiction	2019	48
6	9	Optimized interactive challenge	Colin Buckley	Fantasy	1987	14
7	10	Ergonomic national hub	Samantha Ruiz	Mystery	2015	24
8	11	Secured zero tolerance time-frame	Denise Barnes	Fantasy	1998	35
9	12	Polarized optimal array	Destiny Scott	Non-Fiction	1989	27
10	15	User-friendly motivating strategy	Keith Smith	Non-Fiction	1997	23
11	17	Reduced secondary core	Benjamin Peters	Fantasy	1966	5
12	18	Adaptive 4thgeneration concept	Hector Palmer	Non-Fiction	2021	39
13	19	Progressive asymmetric Internet solution	Sean Miller	Science Fiction	1990	11
14	20	Face-to-face systematic throughput	Teresa Brennan	Non-Fiction	1978	48
15	23	Reverse-engineered context-sensitive hardware	Christina Hernandez	Mystery	1967	38
16	25	Devolved mobile conglomeration	Alexander Bailey	Biography	1984	8
17	26	Multi-channeled multi-tasking capability	Patricia Buck	Science Fiction	1964	21

3. List all customers from Canada:

SQL QUERY:

```
SELECT * FROM CUSTOMERS
WHERE country = 'Canada';
```

RESULT:

	customer_id [PK] integer	name character varying (100)	email character varying (100)	phone character varying (15)	city character varying (50)	country character varying (150)
1	38	Nicholas Harris	christine93@perkins.com	1234567928	Davistown	Canada
2	415	James Ramirez	robert54@hall.com	1234568305	Maxwelltown	Canada
3	468	David Hart	stokesrebecca@gmail.com	1234568358	Thompsonfurt	Canada

4. Show orders placed in November 2023:

SQL QUERY:

```
SELECT * FROM ORDERS
WHERE order_date BETWEEN '2023-11-01' AND '2023-11-30';
```

RESULT:

order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
4	433	343	2023-11-25	7	301.21
19	496	60	2023-11-17	9	316.26
75	291	375	2023-11-30	5	170.75
132	469	333	2023-11-22	7	194.32
137	474	471	2023-11-25	8	363.04
163	207	384	2023-11-23	3	101.76
182	129	293	2023-11-01	7	125.51
200	313	303	2023-11-23	1	6.57
213	325	447	2023-11-17	7	253.75
231	22	384	2023-11-11	1	33.92
245	386	97	2023-11-01	9	411.66
252	405	387	2023-11-15	5	237.10
257	123	403	2023-11-06	1	15.01
288	6	128	2023-11-13	1	24.04
307	368	133	2023-11-17	1	20.96
322	270	112	2023-11-08	2	16.04
344	385	218	2023-11-25	5	26.80
389	485	391	2023-11-18	2	66.84

5. Retrieve the total stock of books available:

SQL QUERY:

```
SELECT SUM(stock) AS TOTAL_STOCK
FROM BOOKS;
```

RESULT:

	total_stock bigint
1	25056

6. Find the details of the most expensive book:

SQL QUERY:

```
SELECT * FROM BOOKS
ORDER BY price DESC
LIMIT 1;
```

RESULT:

	book_id [PK] integer	title character vary	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	340	Proactive system-worthy orchestration	Robert Scott	Mystery	1907	49.98	88

Graph Visualiser

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

SQL QUERY:

```
SELECT * FROM ORDERS
WHERE quantity > 1;
```

RESULT:

	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	1	84	169	2023-05-26	8	188.56
2	2	137	301	2023-01-23	10	216.60
3	3	216	261	2024-05-27	6	85.50
4	4	433	343	2023-11-25	7	301.21
5	5	14	431	2023-07-26	7	136.36
6	6	439	119	2024-10-11	5	249.40
7	7	195	467	2023-10-23	6	82.92
8	8	32	159	2024-05-07	4	144.84
9	9	109	407	2024-01-04	9	379.71
10	10	94	122	2024-07-09	4	123.00
11	12	454	3	2024-06-17	2	31.50
12	13	420	180	2023-06-08	5	125.45
13	14	454	319	2023-08-24	2	85.22
14	15	127	479	2023-01-10	6	229.62
15	16	412	196	2023-10-06	8	53.52

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

SQL QUERY:

```
SELECT * FROM ORDERS
WHERE total_amount > 20;
```

RESULT:

	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	1	84	169	2023-05-26	8	188.56
2	2	137	301	2023-01-23	10	216.60
3	3	216	261	2024-05-27	6	85.50
4	4	433	343	2023-11-25	7	301.21
5	5	14	431	2023-07-26	7	136.36
6	6	439	119	2024-10-11	5	249.40
7	7	195	467	2023-10-23	6	82.92
8	8	32	159	2024-05-07	4	144.84
9	9	109	407	2024-01-04	9	379.71
10	10	94	122	2024-07-09	4	123.00
11	11	131	206	2023-10-16	1	38.01
12	12	454	3	2024-06-17	2	31.50
13	13	420	180	2023-06-08	5	125.45
14	14	454	319	2023-08-24	2	85.22
15	15	127	479	2023-01-10	6	229.62
16	16	412	196	2023-10-06	8	53.52

9. **List all genres available in the Books table**:

SQL QUERY:

```
SELECT DISTINCT genre
FROM BOOKS;
```

RESULT:

	genre character varying (50)
1	Romance
2	Biography
3	Mystery
4	Fantasy
5	Fiction
6	Non-Fiction
7	Science Fiction

10. Find the book with the lowest stock:

SQL QUERY:

```
SELECT * FROM BOOKS
ORDER BY stock ASC
LIMIT 1;
```

RESULT:

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	44	Networked systemic implementation	Ryan Frank	Science Fiction	1965	13.55	0

11. Calculate total revenue generated from all orders:

SQL QUERY:

```
SELECT SUM(total_amount) AS Revenue
FROM ORDERS;
```

RESULT:

	revenue numeric
1	75628.66

12- **Retrieve the total number of books sold for each genre**:

SQL QUERY:

```
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;
```

RESULT:

	genre character varying (50)	total_books_sold bigint
1	Romance	439
2	Biography	285
3	Mystery	504
4	Fantasy	446
5	Fiction	225
6	Non-Fiction	351
7	Science Fiction	447

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

SQL QUERY:

```
SELECT AVG(price) AS Average_Price
FROM BOOKS
WHERE genre = 'Fantasy';
```

RESULT:

	average_price numeric
1	25.9816901408450704