SQL Queries:

Objective 1: Subqueries

1.1: **Find books priced above the average price for their genre**.

```
select
	bookid,
	genrename,
	price
from
	bookstores.books b
where
	price > (
	select
		(cast(avg(price) as decimal(4, 2)))
	from
		bookstores.books b2
	where
		b.genrename = b2.genrename);
```

books 1

select bookid, genrename, price from bookst | Enter a SQL expression to filter results (use Ctrl+Space)

	bookid	genrename	price
1	1	Fantasy	52.37
2	5	Historical Fiction	55.53
3	6	Fantasy	44.81
4	9	Autobiography	54.29
5	10	Religion	27.62
6	11	Food and Drink	38.77
7	12	Poetry	51.77
8	13	Fiction	39.72
9	17	Spirituality	55.65
10	18	Historical Fiction	39.01
11	19	History	56.63

Refresh Save Cancel Export data 200 200+

200 row(s) fetched - 0.078s (0.001s fetch), on 2024-11-27 at 20:17:03

1.2: **Identify customers with purchases exceeding a specific value using subqueries.**

```
select
	c.customerid,
	c.name
from
	bookstores.customers c
where
	(
	select
		sum(o2.price * o2.quantity)
	from
		bookstores.orderdetails o2
	join bookstores.orders o on
		o2.orderid = o.orderid
	where
		o.customerid = c.customerid )
> 100 ;
```

customers 1

select c.customerid, c.name from bookstores | Enter a SQL expression to filter results (use Ctrl+Space)

	customerid	name
1	1	John Smith
2	2	Maria Gonzalez
3	4	Emily Davis
4	6	Aiko Tanaka
5	7	Robert Wilson
6	8	Fatima Ahmed
7	13	Christopher Lee
8	16	Sofia Rossi
9	19	Paul Young

Refresh Save Cancel Export data 200 47

2. **Display each genre with the total stock of books using a subquery.**

```
select
        b.genrename,
        (
        select
                sum(b2.quantity)
        from
                bookstores.books b2
        where
                b2.genrename = b.genrename) as total_stock
from
        bookstores.books b
group by
        b.genrename ;
```

books 1

select b.genrename, (select sum(b2.quantity) | Enter a SQL expression to filter results (use Ctrl+Space)

	genrename	total_stock
1	Adult Fiction	12
2	Art	50
3	Autobiography	59
4	Biography	56
5	Business	118
6	Childrens	217
7	Christian	6
8	Christian Fiction	47
9	Classics	135
10	Contemporary	22
11	Crime	15
12	Erotica	9

Refresh Save Cancel Export data 200 44

BDT en Writable Smart Insert 25 : 1 [177] Sel: 177 | 5

Objective 2: Common Table Expressions (CTEs):

2.1: **List the top 5 customers with the highest spending in a specific year.**

```
with totalspending as (
select
       o.customerid,
       c.name,
       sum(price * quantity) as total_spending
from
       bookstores.orderdetails o2
join bookstores.orders o on
       o2.orderid = o.orderid
join bookstores.customers c on
       o.customerid = c.customerid
where
       year(o.orderdate) = 2024
group by
       o.customerid
             )
select
       customerid,
       name,
       total_spending
from
       totalspending
order by
       total_spending desc
limit 5 ;
```

orders(+) 1

with totalspending as (select o.customerid, (| Enter a SQL expression to filter results (use Ctrl+Space)

	customerid	name	total_spending
1	82	Avery Wilson	297.4
2	53	Dylan Fisher	286.75
3	19	Paul Young	279.25
4	34	Lisa Nelson	277.3
5	2	Maria Gonzalez	273.25

Refresh Save Cancel Export data 200 5

BDT | en | Writable | Smart Insert | 17 : 13 : 545 | Sel: 0 | 0

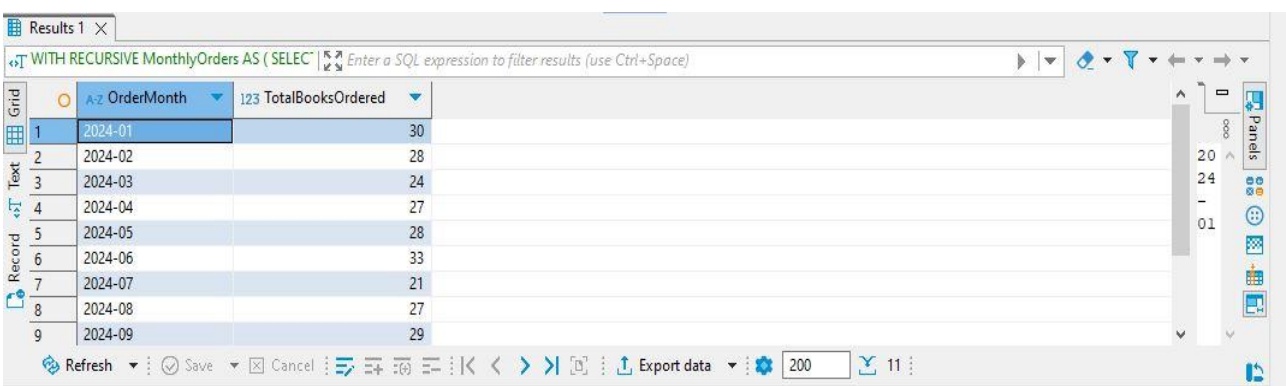
2.2: **Track the total number of books ordered monthly using a recursive CTE.**

```sql
WITH RECURSIVE MonthlyOrders AS (
    -- Anchor: Start with the earliest order date
    SELECT
        MIN(DATE_FORMAT(OrderDate, '%Y-%m-01')) AS MonthStart
    FROM
        Bookstores.Orders

    UNION ALL

    -- Recursive step: Add one month at a time
    SELECT
        DATE_ADD(MonthStart, INTERVAL 1 MONTH)
    FROM
        MonthlyOrders
    WHERE
        MonthStart < (SELECT MAX(DATE_FORMAT(OrderDate, '%Y-%m-01')) FROM
Bookstores.Orders)
)

-- Step 2: Calculate the total number of books ordered for each month
SELECT
    DATE_FORMAT(MonthStart, '%Y-%m') AS OrderMonth,
    COALESCE(SUM(od.Quantity), 0) AS TotalBooksOrdered
FROM
    MonthlyOrders mo
LEFT JOIN
    Bookstores.Orders o ON DATE_FORMAT(o.OrderDate, '%Y-%m') =
DATE_FORMAT(mo.MonthStart, '%Y-%m')
LEFT JOIN
    Bookstores.OrderDetails od ON o.OrderID = od.OrderID
GROUP BY
    MonthStart
ORDER BY
    MonthStart;
```



	OrderMonth	TotalBooksOrdered
1	2024-01	30
2	2024-02	28
3	2024-03	24
4	2024-04	27
5	2024-05	28
6	2024-06	33
7	2024-07	21
8	2024-08	27
9	2024-09	29

Objective 3: JOIN Operations

3.1: **Display a list of books with their authors, genres, and stock quantity.**

```
select
	a.authorid,
	a.authorname,
	b.genrename,
	b.quantity as stock_quantity
from
	bookstores.authors a
join bookstores.books b on
	a.authorid = b.authorid ;
```

authors(+) 1

select a.authorid, a.authorname, b.genrenam | Enter a SQL expression to filter results (use Ctrl+Space)

	authorid	authorname	genrename	stock_quantity
1	1	A.L. Jackson	Romance	16
2	2	Abbi Glines	Young Adult	1
3	3	Adam Clymer	History	7
4	4	Adam Higginbotham	Nonfiction	1
5	5	Agatha Christie	Mystery	6
6	5	Agatha Christie	Mystery	4
7	6	Aisha Saeed	Young Adult	3
8	7	Alain de Botton	Fiction	1
9	8	Alan Turing	Autobiography	3
10	9	Alexander McCall Smith	Mystery	9
11	10	Ali Benjamin	Childrens	18

Refresh | Save | Cancel | Export data | 200 | 637

637 row(s) fetched - 0.021s (0.002s fetch), on 2024-11-28 at 13:17:32

BDT | en | Writable | Smart Insert | 58 : 29 : 1749 | Sel: 0 | 0

3.2: **List orders with the total cost, including book titles and quantities.**

```
select
	o.orderid,
	b.title,
	o.quantity,
	o.Orderamount as total_cost
from
	bookstores.orderdetails o
join bookstores.books b on
	o.Bookid = b.bookid ;
```

orderdetails(+) 1

select o.orderid, b.title, o.quantity, o.Orderan | Enter a SQL expression to filter results (use Ctrl+Space)

orderid	title	quantity	total_cost
1	Fruits Basket, Vol. 6 (Fruits Basket #6)	3	155.25
2	Scott Pilgrim's Precious Little Life (Scott Pilgrim #1)	5	273.25
3	Close to You	2	43.88
4	The Lonely City: Adventures in the Art of Being Alone	4	156.96
5	A la Mode: 120 Recipes in 60 Pairings: Pies, Tarts, Cakes, Crisps, and More	1	38.77
6	Misery	5	170
7	Wuthering Heights	3	173.1
8	Batman: The Dark Knight Returns (Batman)	4	194.52
9	The No. 1 Ladies' Detective Agency (No. 1 Ladies' Detective Agency #1)	2	49.14

Refresh Save Cancel Export data 200 100

BDT | en | Writable | Smart Insert | 13 : 1 : 251 | Sel: 0 | 0

Objective 4: Handling NULLs

4.1: **Show all customers with NULL if they have no orders.**

```
select
       c.customerid,
       c.email,
       c.city,
       o.orderdate
from
       bookstores.customers c
left join bookstores.orders o on
       c.customerid = o.customerid ;
```

N.B: Orederid, Orderdate has no null value in my data sheet.

customers(+) 1

select c.customerid, c.email, c.city, o.orderda | Enter a SQL expression to filter results (use Ctrl+Space)

customerid	email	city	orderdate
1	john.smith@email.com	New York	2024-01-01
2	maria.gonzalez@email.com	Madrid	2024-02-01
3	ahmed.khan@email.com	Lahore	2024-03-01
4	emily.davis@email.com	Sydney	2024-04-01
5	william.johnson@email.com	Toronto	2024-05-01
6	aiko.tanaka@email.com	Tokyo	2024-06-01
7	robert.wilson@email.com	London	2024-07-01
8	fatima.ahmed@email.com	Cairo	2024-08-01

Refresh Save Cancel Export data 200 100

100 row(s) fetched - 0.008s, on 2024-11-28 at 15:01:43

BDT | en | Writable | Smart Insert | 12 : 34 : 227 | Sel: 0 | 0

4.2: List genres with the number of books available, showing zero for genres with no books.

```
select
     g.genreid,
     g.genrename,
     coalesce(sum(quantity), 0) as number_of_books
from
     bookstores.genres g
```

```
join bookstores.books b on
        g.genrename = b.genrename
group by
        g.genrename
order by

        number_of_books ;
```

genres 1

select g.genreid, g.genrename, sum(quantity | Enter a SQL expression to filter results (use Ctrl+Space)

	genreid	genrename	number_of_books
1	44	Paranormal	[NULL]
2	28	Parenting	3
3	24	Christian	6
4	42	Novels	8
5	31	Erotica	9
6	36	Adult Fiction	12
7	43	Crime	15
8	17	Historical	18
9	41	Contemporary	22

Refresh Save Cancel Export data 200 44

BDT en Writable Smart Insert 10 : 26 : 306 Sel: 0 | 0

genres 1

select g.genreid, g.genrename, coalesce(sun | Enter a SQL expression to filter results (use Ctrl+Space)

	genreid	genrename	number_of_books
1	44	Paranormal	0
2	28	Parenting	3
3	24	Christian	6
4	42	Novels	8
5	31	Erotica	9
6	36	Adult Fiction	12
7	43	Crime	15
8	17	Historical	18
9	41	Contemporary	22

Refresh Save Cancel Export data 200 44

BDT en Writable Smart Insert 16 : 20 : 326 Sel: 0 | 0

Objective 5: Query Optimization

1. **Create and optimize queries using indexes and EXPLAIN statements.**
2. **Rewrite the query for the top 5 highest-priced books to improve performance.**

Query creation without optimization:

```
select
        bookid,
        title,
        price
from
        bookstores.books b
order by
        price desc
        limit 5 ;
```

use Explain statements:

```
explain select
	bookid,
	title,
	price
from
	bookstores.books b
order by
	price desc
limit 5 ;
```

Results 1

explain select bookid, title, price from books | Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	A-Z select_type	A-Z table	A-Z partitions	A-Z type	A-Z possible_keys	A-Z key	A-Z key_len	A-Z ref	123 rov
1	1	SIMPLE	b	[NULL]	ALL	[NULL]	[NULL]	[NULL]	[NULL]	

Results 1

explain select bookid, title, price from books | Enter a SQL expression to filter results (use Ctrl+Space)

	le	A-Z partitions	A-Z type	A-Z possible_keys	A-Z key	A-Z key_len	A-Z ref	123 rows	123 filtered	A-Z Extra
1		[NULL]	ALL	[NULL]	[NULL]	[NULL]	[NULL]	637	100	Using filesort

Index add on price:

```
CREATE INDEX idx_books_price ON bookstores.books(price) ;

explain select
	bookid,
	title,
	price
from
	bookstores.books b
order by
	price desc
limit 5 ;
```

Results 1

explain select bookid, title, price from books! | Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	A-Z select_type	A-Z table	A-Z partitions	A-Z type	A-Z possible_keys	A-Z key	A-Z key_len	A-Z ref	123
1	1	SIMPLE	b	[NULL]	index	[NULL]	idx_books_price	9	[NULL]	

****Before Indexing Price: Explain Analyze :**

-> Limit: 5 row(s) (cost=65.2 rows=5) (actual time=0.578..0.581 rows=5 loops=1)

-> Sort: b.price DESC, limit input to 5 row(s) per chunk (cost=65.2 rows=637) (actual time=0.576..0.578 rows=5 loops=1)

-> Table scan on b (cost=65.2 rows=637) (actual time=0.099..0.446 rows=637 loops=1)

****After Indexing Price: Explain Analyze:**

-> Limit: 5 row(s) (cost=0.0157 rows=5) (actual time=0.0801..0.0828 rows=5 loops=1)

-> Index scan on b using idx_books_price (reverse) (cost=0.0157 rows=5) (actual time=0.0792..0.0815 rows=5 loops=1)