DE235 Soumyadeep Sinha
WEEK 1 - DAY 3 - 30-10-2024

SQL CLASSROOM ACTIVITY PRACTICE

---

1. Database Creation and Setup

- **Drop and Create Database**: Ensure a fresh database by dropping it if it exists and then creating it.

```
IF EXISTS (SELECT * FROM sys.databases WHERE name = 'Library')
BEGIN
   DROP DATABASE Library;
END;
CREATE DATABASE Library;
USE Library;
```

2. Creating the books Table

- Define Fields and Primary Key: Set up columns for book information and ensure each book has a unique identifier.

```
CREATE TABLE books (
   book_id INT IDENTITY(1,1) PRIMARY KEY,
   title VARCHAR(255) NOT NULL,
   author VARCHAR(255),
   genre VARCHAR(100),
   published_year INT,
   isbn VARCHAR(20) NOT NULL UNIQUE,
   price DECIMAL(10, 2)
);
```

3. Inserting Book Records

- **Add Sample Data**: Insert multiple records into `books`.

```sql
INSERT INTO books (title, author, genre, published_year, isbn, price) VALUES
('Pride and Prejudice', 'Jane Austen', 'Fiction', 1813, '1112223334445', 12.99),
('1984', 'George Orwell', 'Dystopian', 1949, '2223334445556', 15.99),
('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, '3334445556667', 10.99);

select * from books;
```

110 %

Results | Messages

	book_id	title	author	genre	published_year	isbn	price
1	1	Pride and Prejudice	Jane Austen	Fiction	1813	1112223334445	12.99
2	2	1984	George Orwell	Dystopian	1949	2223334445556	15.99
3	3	To Kill a Mockingbird	Harper Lee	Fiction	1960	3334445556667	10.99

4. Data Integrity Constraints

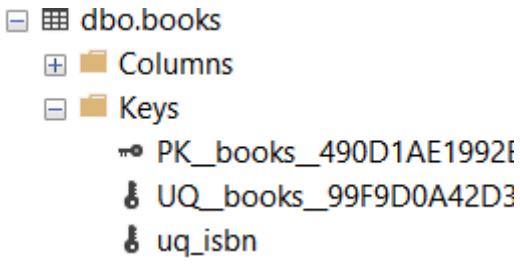
- **Enforce Minimum Price**: Ensure book prices cannot be lower than $5.

```sql
ALTER TABLE books ADD CONSTRAINT chk_price CHECK (price >= 5.00);
```

- dbo.books
  - Columns
  - Keys
  - Constraints
    - chk_price
  - Triggers
  - Indexes
  - Statistics

- **ISBN Uniqueness**: Ensure ISBNs are unique.

```
ALTER TABLE books ADD CONSTRAINT uq_isbn UNIQUE (isbn);
```



5. String Manipulation Functions

- **Convert Title to Uppercase**
- **Concatenate Title and Author**
- **Extract Genre Prefix** (using `SUBSTRING` for the first 3 characters)

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, '3334445556667', 10.99);

--select * from books;

--ALTER TABLE books ADD CONSTRAINT chk_price CHECK (price >= 5.00);

--ALTER TABLE books ADD CONSTRAINT uq_isbn UNIQUE (isbn);

SELECT UPPER(title) AS title_upper FROM books;
SELECT CONCAT(title, ' by ', author) AS full_details FROM books;
SELECT title, SUBSTRING(genre, 1, 3) AS genre_prefix FROM books;
```

110 %

Results | Messages

	title_upper
1	PRIDE AND PREJUDICE
2	1984
3	TO KILL A MOCKINGBIRD

	full_details
1	Pride and Prejudice by Jane Austen
2	1984 by George Orwell
3	To Kill a Mockingbird by Harper Lee

	title	genre_prefix
1	Pride and Prejudice	Fic
2	1984	Dys
3	To Kill a Mockingbird	Fic

Query executed successfully. | DEEP-TUF-FX504\SQLEXPRESS (... | DEEP-TUF-FX504\deeps (66) | Library | 00:00:00 | 9 rows

6. Date Functions and Age Calculations

- **Extract Year from Published Date**
- **Calculate Book Age**

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--ALTER TABLE books ADD CONSTRAINT chk_price CHECK (price >= 5.00);

--ALTER TABLE books ADD CONSTRAINT uq_isbn UNIQUE (isbn);

--SELECT UPPER(title) AS title_upper FROM books;
--SELECT CONCAT(title, ' by ', author) AS full_details FROM books;
--SELECT title, SUBSTRING(genre, 1, 3) AS genre_prefix FROM books;

SELECT title, published_year, YEAR(GETDATE()) - published_year AS book_age
FROM books;
SELECT title, DATEDIFF(YEAR, DATEFROMPARTS(published_year, 1, 1), GETDATE()) AS age_in_years
FROM books;
```

110 %

Results | Messages

	title	published_year	book_age
1	Pride and Prejudice	1813	211
2	1984	1949	75
3	To Kill a Mockingbird	1960	64

	title	age_in_years
1	Pride and Prejudice	211
2	1984	75
3	To Kill a Mockingbird	64

Query executed successfully. DEEP-TUF-FX504\SQLEXPRESS (... DEEP-TUF-FX504\deeps (66) Library 00:00:00 6 rows

7. Aggregation and Summaries

- **Count Books by Genre**
- **Calculate Average and Total Prices**

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--SELECT CONCAT(title, ' by ', author) AS full_details FROM books;
--SELECT title, SUBSTRING(genre, 1, 3) AS genre_prefix FROM books;

--SELECT title, published_year, YEAR(GETDATE()) - published_year AS book_age
--FROM books;
--SELECT title, DATEDIFF(YEAR, DATEFROMPARTS(published_year, 1, 1), GETDATE()) AS age_in_years
--FROM books;

SELECT genre, COUNT(*) AS book_count FROM books
GROUP BY genre;
SELECT COUNT(*) AS book_count, AVG(price) AS average_price, SUM(price) AS total_price
FROM books;
```

110 %

Results | Messages

	genre	book_count
1	Dystopian	1
2	Fiction	2

	book_count	average_price	total_price
1	3	13.323333	39.97

Query executed successfully. | DEEP-TUF-FX504\SQLEXPRESS (... | DEEP-TUF-FX504\deeps (66) | Library | 00:00:00 | 3 rows

8. Creating the `members` Table with Foreign Key

- **Define Member Table with Foreign Key**: Links each borrowed book to a library member.

```sql
CREATE TABLE members (
    member_id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255),
    phone VARCHAR(50),
    email VARCHAR(100) UNIQUE
);
```

```sql
CREATE TABLE borrowed_books (
    borrow_id INT IDENTITY(1,1) PRIMARY KEY,
    member_id INT NOT NULL,
    book_id INT NOT NULL,
    borrow_date DATE,
    due_date DATE,
    FOREIGN KEY (member_id) REFERENCES members(member_id) ON DELETE CASCADE,
    FOREIGN KEY (book_id) REFERENCES books(book_id) ON DELETE CASCADE
);
```

```sql
INSERT INTO members (name, address, phone, email) VALUES
('Alice Johnson', '123 Maple St', '555-0101', 'alice.johnson@example.com'),
('Bob Smith', '456 Oak St', '555-0102', 'bob.smith@example.com'),
('Carol Williams', '789 Pine St', '555-0103', 'carol.williams@example.com'),
('David Brown', '321 Elm St', '555-0104', 'david.brown@example.com'),
('Eve Davis', '654 Cedar St', '555-0105', 'eve.davis@example.com');

INSERT INTO borrowed_books (member_id, book_id, borrow_date, due_date) VALUES
(1, 1, '2024-10-01', '2024-10-15'),
(2, 2, '2024-10-02', '2024-10-16'),
(3, 1, '2024-10-05', '2024-10-20'),
(4, 3, '2024-10-08', '2024-10-22'),
(5, 2, '2024-10-10', '2024-10-24'),
(1, 3, '2024-10-12', '2024-10-26');
```

9. Joining Tables

- **Inner Join**: Show members with books they borrowed.

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--(2, 2, '2024-10-02', '2024-10-16'),
--(3, 1, '2024-10-05', '2024-10-20'),
--(4, 3, '2024-10-08', '2024-10-22'),
--(5, 2, '2024-10-10', '2024-10-24'),
--(1, 3, '2024-10-12', '2024-10-26');


SELECT m.name, b.title, bb.borrow_date
FROM members m
JOIN borrowed_books bb ON m.member_id = bb.member_id
JOIN books b ON bb.book_id = b.book_id;
```

110 %

Results | Messages

	name	title	borrow_date
1	Alice Johnson	Pride and Prejudice	2024-10-01
2	Bob Smith	1984	2024-10-02
3	Carol Williams	Pride and Prejudice	2024-10-05
4	David Brown	To Kill a Mockingbird	2024-10-08
5	Eve Davis	1984	2024-10-10
6	Alice Johnson	To Kill a Mockingbird	2024-10-12

Query executed successfully. | DEEP-TUF-FX504\SQLEXPRESS (... | DEEP-TUF-FX504\deeps (66) | Library | 00:00:00 | 6 rows

- **Left Join**: Show all members, even if they haven't borrowed any books.

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--(5, 2, '2024-10-10', '2024-10-24'),
--(1, 3, '2024-10-12', '2024-10-26');


--SELECT m.name, b.title, bb.borrow_date
--FROM members m
--JOIN borrowed_books bb ON m.member_id = bb.member_id
--JOIN books b ON bb.book_id = b.book_id;

SELECT m.member_id, m.name, b.title
FROM members m
LEFT JOIN borrowed_books bb ON m.member_id = bb.member_id
LEFT JOIN books b ON bb.book_id = b.book_id;
```

110 %

Results | Messages

	member_id	name	title
1	1	Alice Johnson	Pride and Prejudice
2	1	Alice Johnson	To Kill a Mockingbird
3	2	Bob Smith	1984
4	3	Carol Williams	Pride and Prejudice
5	4	David Brown	To Kill a Mockingbird
6	5	Eve Davis	1984

Query executed successfully. DEEP-TUF-FX504\SQLEXPRESS (... DEEP-TUF-FX504\deeps (66) Library 00:00:00 6 rows

10. Books Borrowed per Member

- **Count Books Borrowed by Each Member**:

prcticeQueries.sql...F-FX504\deeps (66))*

```sql
--JOIN borrowed_books bb ON m.member_id = bb.member_id
--JOIN books b ON bb.book_id = b.book_id;

--SELECT m.member_id, m.name, b.title
--FROM members m
--LEFT JOIN borrowed_books bb ON m.member_id = bb.member_id
--LEFT JOIN books b ON bb.book_id = b.book_id;

SELECT m.member_id, m.name, COUNT(bb.book_id) AS books_borrowed
FROM members m
LEFT JOIN borrowed_books bb ON m.member_id = bb.member_id
GROUP BY m.member_id, m.name;
```

110 %

Results | Messages

	member_id	name	books_borrowed
1	1	Alice Johnson	2
2	2	Bob Smith	1
3	3	Carol Williams	1
4	4	David Brown	1
5	5	Eve Davis	1

Query executed successfully. | DEEP-TUF-FX504\SQLEXPRESS (... | DEEP-TUF-FX504\deeps (66) | Library | 00:00:00 | 5 rows