INFO 6210 Data Management and Database Design

Library System

Group 6 : Shoutai Ma
Yongji Shen
Zhiming Xue
Zunyu Liu
Zijian Zhao

Content List

1.Project Objectives

2.ERD

3.Database Implementation

4.Views Report

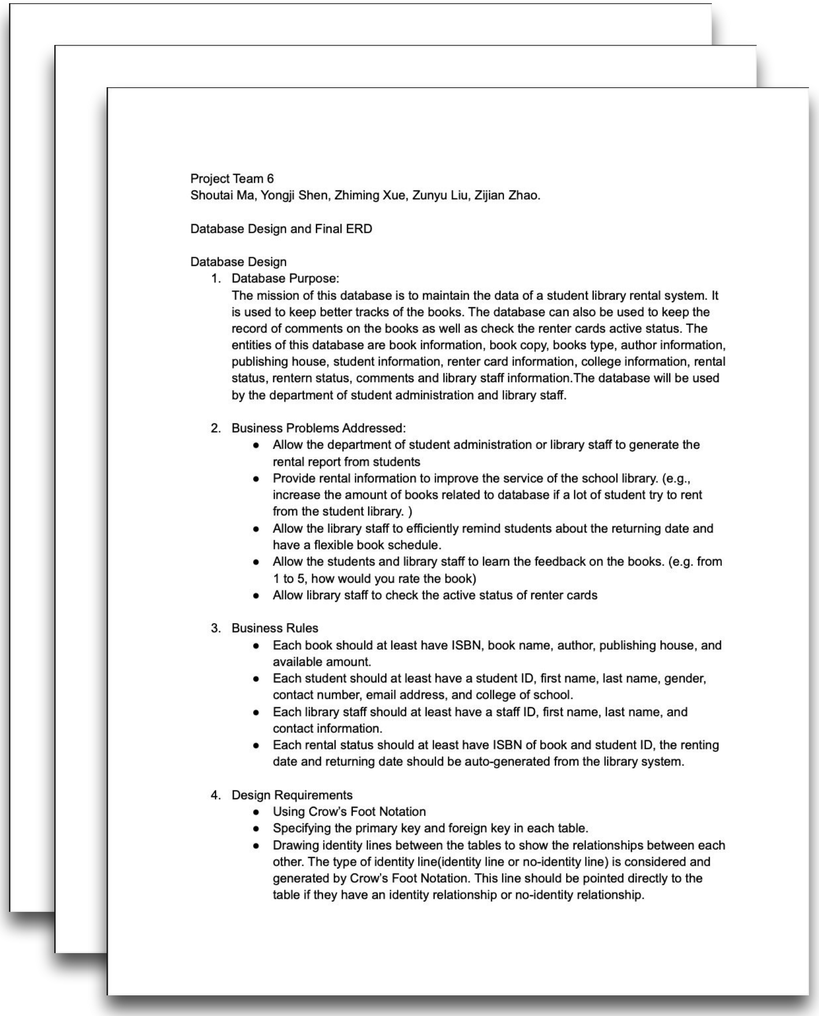
5.Table-Level Constraint

6.Data Encryption

7.Reports & Visualization

Project Objectives

- Maintain the data of a student library rental system.
- Track the rental information of each book.
- Keep the record of comments on the books
- Check the students' cards status.
- The database will be used by the department of student administration and library staff.



Project Team 6
Shoutai Ma, Yongji Shen, Zhiming Xue, Zunyu Liu, Zijian Zhao.

Database Design and Final ERD

Database Design

1. Database Purpose:
   The mission of this database is to maintain the data of a student library rental system. It is used to keep better tracks of the books. The database can also be used to keep the record of comments on the books as well as check the renter cards active status. The entities of this database are book information, book copy, books type, author information, publishing house, student information, renter card information, college information, rental status, rentern status, comments and library staff information.The database will be used by the department of student administration and library staff.

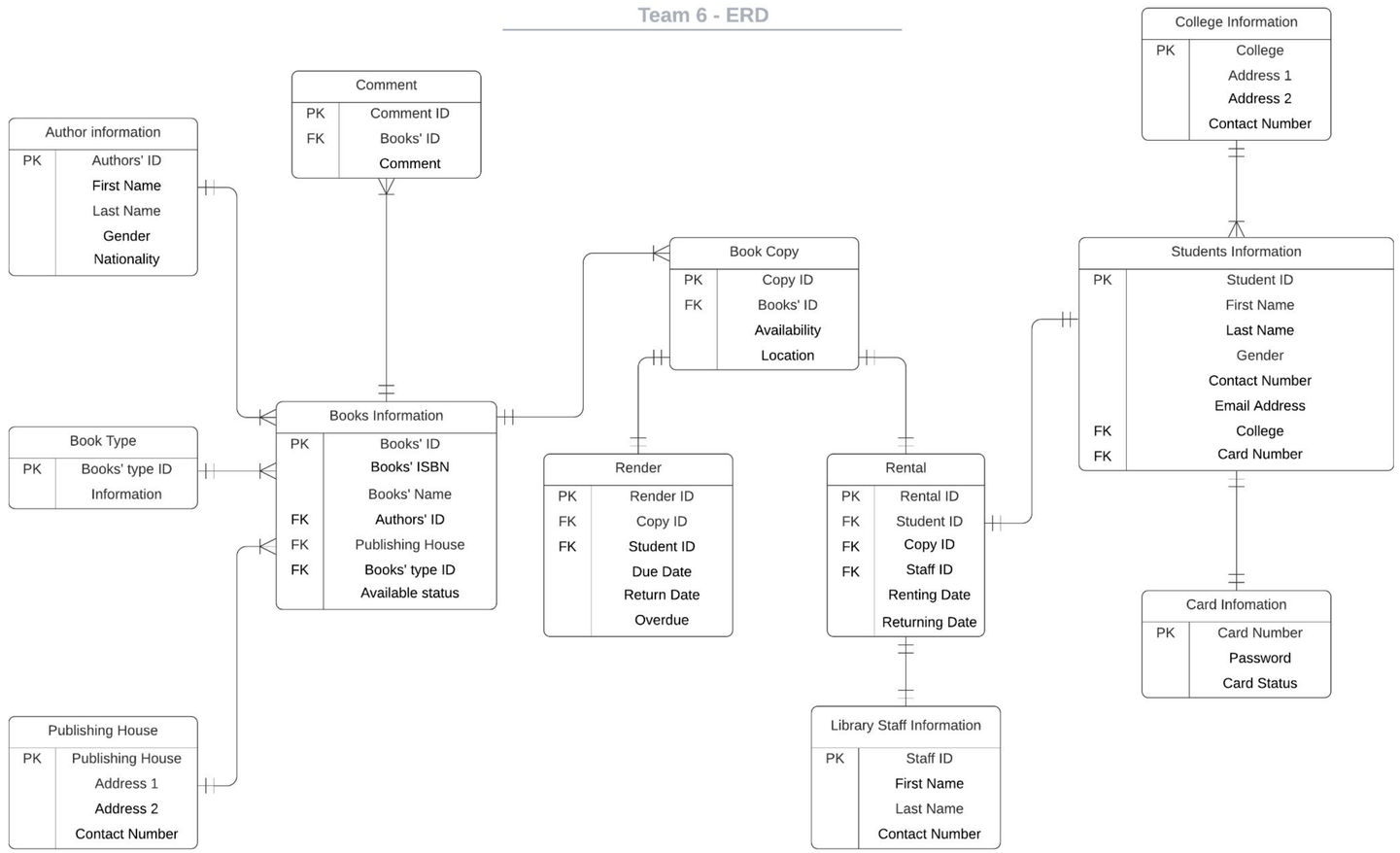
2. Business Problems Addressed:
   - Allow the department of student administration or library staff to generate the rental report from students
   - Provide rental information to improve the service of the school library. (e.g., increase the amount of books related to database if a lot of student try to rent from the student library.)
   - Allow the library staff to efficiently remind students about the returning date and have a flexible book schedule.
   - Allow the students and library staff to learn the feedback on the books. (e.g. from 1 to 5, how would you rate the book)
   - Allow library staff to check the active status of renter cards

3. Business Rules
   - Each book should at least have ISBN, book name, author, publishing house, and available amount.
   - Each student should at least have a student ID, first name, last name, gender, contact number, email address, and college of school.
   - Each library staff should at least have a staff ID, first name, last name, and contact information.
   - Each rental status should at least have ISBN of book and student ID, the renting date and returning date should be auto-generated from the library system.

4. Design Requirements
   - Using Crow's Foot Notation
   - Specifying the primary key and foreign key in each table.
   - Drawing identity lines between the tables to show the relationships between each other. The type of identity line(identity line or no-identity line) is considered and generated by Crow's Foot Notation. This line should be pointed directly to the table if they have an identity relationship or no-identity relationship.

ERD

- There are a total of 12 entities.
- Divided into three schemas: Book, Rental, and Student.
- Book schema contains Comment, BookInformation, AuthorInformation, BookType, and PublishingHouse.
- Rental schema contains BookCopy, Rental, Return and LibraryStaff.
- Student schema contains CardInformation, StudentInformation, and CollegeInformation.



Database Implementation

Create Schema - Student, then create the following tables and set its data type, PK and FK.

- StudentInformation
- CardInformation
- CollegeInformation

```
CREATE TABLE Student.StudentInformation
        (
        StudentID int NOT NULL PRIMARY KEY,
        FirstName varchar(40) NOT NULL,
        LastName varchar(40) NOT NULL,
        Gender varchar(40) NOT NULL,
        ContactNumber int NOT NULL,
        EmailAddress varchar(40) NOT NULL,
        College varchar(40) NOT NULL REFERENCES Student.CollegeInformation(College),
        CardNumber int NOT NULL REFERENCES Student.CardInformation(CardNumber)
        );
CREATE TABLE Student.CardInformation
        (
        CardNumber int NOT NULL PRIMARY KEY,
        Password varchar(20) NOT NULL,
        CardStatus varchar(20) NOT NULL
        );
CREATE TABLE Student.CollegeInformation
        (
        College varchar(40) NOT NULL PRIMARY KEY,
        Address1 varchar(60) NOT NULL,
        Address2 varchar(60) NOT NULL,
        ContactNumber int NOT NULL
        );
```

Database Implementation

Create Schema - Book, then create the following tables and set its data type, PK and FK.

- Comment
- BookInformation
- AuthorInformation
- BookType
- PublishingHose

```
CREATE TABLE Book.Comment
        (
        CommentID int NOT NULL PRIMARY KEY,
        BooksID int NOT NULL REFERENCES Book.BookInformation(BooksID),
        Comment varchar(40)
        );
CREATE TABLE Book.BookInformation
        (
        BooksID int NOT NULL PRIMARY KEY,
        BooksISBN varchar(255) NOT NULL,
        BooksName varchar(255) NOT NULL,
        AuthorsID int NOT NULL REFERENCES Book.AuthorInformation(AuthorsID),
        PublishingHouse varchar(60) NOT NULL REFERENCES Book.PublishingHouse(PublishingHouse),
        BooksTypeID int NOT NULL REFERENCES Book.BookType(BooksTypeID),
        AvailableStatus varchar(40) NOT NULL
        );
CREATE TABLE Book.AuthorInformation
        (
        AuthorsID int NOT NULL PRIMARY KEY,
        FirstName varchar(40) NOT NULL,
        LastName varchar(40) NOT NULL,
        Gender varchar(40) NOT NULL,
        Nationality varchar(40) NOT NULL
        );
CREATE TABLE Book.BookType
        (
        BooksTypeID int NOT NULL PRIMARY KEY,
        Information varchar(40) NOT NULL
        );
CREATE TABLE Book.PublishingHouse
        (
        PublishingHouse varchar(60) NOT NULL PRIMARY KEY,
        Address1 varchar(60) NOT NULL,
        Address2 varchar(60) NOT NULL,
        ContactNumber int NOT NULL
        );
```

Database Implementation

Create Schema - Rental, then create the following tables and set its data type, PK and FK.

- BookCopy
- LibraryStaff
- Render
- Rental

Computed Column:

The DueDate is calculated to provide convenience to set a deadline for returning a book.

The OverDue is calculated to provide convenience to keep track of the student's behavior.

```
CREATE TABLE Rental.BookCopy
        (
        CopyID int NOT NULL PRIMARY KEY,
        BooksID int NOT NULL REFERENCES Book.BookInformation(BooksID),
        Availability int,
        Location varchar(40) NOT NULL
        );
CREATE TABLE Rental.LibraryStaff
        (
        StaffID int NOT NULL PRIMARY KEY,
        FirstName varchar(40) NOT NULL,
        LastName varchar(40) NOT NULL,
        ConcatNumber int NOT NULL
        );

CREATE TABLE Rental.Render
        (
        ReturnID int NOT NULL IDENTITY PRIMARY KEY,
        CopyID int NOT NULL REFERENCES Rental.BookCopy(CopyID),
        StudentID int NOT NULL REFERENCES Student.StudentInformation(StudentID),
        DueDate DATE not null,
        RenderDate DATE,
        OverDue as DATEDIFF(day,DueDate,isnull(RenderDate,DueDate))
        );

CREATE TABLE Rental.Rental
        (
        RentalID int NOT NULL IDENTITY PRIMARY KEY,
        StudentID int NOT NULL REFERENCES Student.StudentInformation(StudentID),
        CopyID int NOT NULL REFERENCES Rental.BookCopy(CopyID),
        StaffID int NOT NULL REFERENCES Rental.LibraryStaff(StaffID),
        RentalDate date not null,
        DueDate as (DATEADD(day,30,RentalDate))
        );
```

Database Implementation

Stored Procedures/Trigger

```
create trigger Rental.updateDueDate
on Rental.Rental
after INSERT
AS
BEGIN
    insert into Rental.Render (CopyID,StudentID,DueDate)
    select CopyID, StudentID, DueDate from inserted;
END

drop trigger if exists Rental.updateDueDate;
```

Create a Trigger - Rental.updateDueDate
The trigger helps to insert CopyID, StudentID, DueDate while Rental.Rental is inserting data

Example

Rental.Rental

	RentalID	StudentID	CopyID	StaffID	RentalDate	DueDate
1	1	2,102	52	112	2020-01-02	2020-02-01
2	2	2,102	11	112	2020-01-02	2020-02-01
3	3	2,101	61	114	2020-01-05	2020-02-04
4	4	2,101	71	114	2020-01-05	2020-02-04
5	5	2,101	21	114	2020-01-05	2020-02-04
6	6	2,101	51	114	2020-01-05	2020-02-04
7	7	2,105	62	116	2020-01-06	2020-02-05
8	8	2,105	31	116	2020-01-06	2020-02-05
9	9	2,106	63	116	2020-01-06	2020-02-05
10	10	2,106	33	116	2020-01-06	2020-02-05
11	11	2,110	21	117	2020-03-15	2020-04-14
12	12	2,104	32	119	2020-03-17	2020-04-16
13	13	2,102	92	115	2020-08-05	2020-09-04
14	14	2,108	61	114	2021-01-05	2021-02-04

Rental.Render

	ReturnID	CopyID	StudentID	DueDate	RenderDate	OverDue
1	1	61	2,108	2021-02-04	2021-04-16	71
2	2	92	2,102	2020-09-04	[NULL]	0
3	3	32	2,104	2020-04-16	[NULL]	0
4	4	21	2,110	2020-04-14	[NULL]	0
5	5	33	2,106	2020-02-05	[NULL]	0
6	6	63	2,106	2020-02-05	[NULL]	0
7	7	31	2,105	2020-02-05	[NULL]	0
8	8	62	2,105	2020-02-05	[NULL]	0
9	9	51	2,101	2020-02-04	[NULL]	0
10	10	21	2,101	2020-02-04	[NULL]	0
11	11	71	2,101	2020-02-04	[NULL]	0
12	12	61	2,101	2020-02-04	[NULL]	0
13	13	11	2,102	2020-02-01	[NULL]	0
14	14	52	2,102	2020-02-01	[NULL]	0

Database Implementation

Stored Procedures/Trigger

```
create PROCEDURE Rental.setReturnDate
@CopyID int,
@StudentID int
AS
BEGIN
    update Rental.Render
    set RenderDate = GETDATE()
    where CopyID = @CopyID and StudentID = @StudentID
END


exec Rental.setReturnDate  61, 2108;
```

Create a Stored Procedure - Rental.setReturnDate
This Procedure will update the return date in the Rental.Render

Example

Rental.Rental

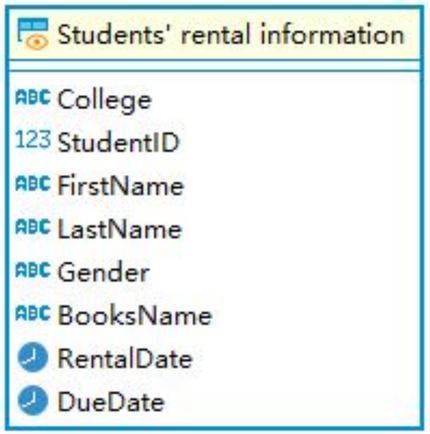
	RentalID	StudentID	CopyID	StaffID	RentalDate	DueDate
1	1	2,102	52	112	2020-01-02	2020-02-01
2	2	2,102	11	112	2020-01-02	2020-02-01
3	3	2,101	61	114	2020-01-05	2020-02-04
4	4	2,101	71	114	2020-01-05	2020-02-04
5	5	2,101	21	114	2020-01-05	2020-02-04
6	6	2,101	51	114	2020-01-05	2020-02-04
7	7	2,105	62	116	2020-01-06	2020-02-05
8	8	2,105	31	116	2020-01-06	2020-02-05
9	9	2,106	63	116	2020-01-06	2020-02-05
10	10	2,106	33	116	2020-01-06	2020-02-05
11	11	2,110	21	117	2020-03-15	2020-04-14
12	12	2,104	32	119	2020-03-17	2020-04-16
13	13	2,102	92	115	2020-08-05	2020-09-04
14	14	2,108	61	114	2021-01-05	2021-02-04

Rental.Render

	ReturnID	CopyID	StudentID	DueDate	RenderDate	OverDue
1	1	61	2,108	2021-02-04	2021-04-16	71
2	2	92	2,102	2020-09-04	[NULL]	0
3	3	32	2,104	2020-04-16	[NULL]	0
4	4	21	2,110	2020-04-14	[NULL]	0
5	5	33	2,106	2020-02-05	[NULL]	0
6	6	63	2,106	2020-02-05	[NULL]	0
7	7	31	2,105	2020-02-05	[NULL]	0
8	8	62	2,105	2020-02-05	[NULL]	0
9	9	51	2,101	2020-02-04	[NULL]	0
10	10	21	2,101	2020-02-04	[NULL]	0
11	11	71	2,101	2020-02-04	[NULL]	0
12	12	61	2,101	2020-02-04	[NULL]	0
13	13	11	2,102	2020-02-01	[NULL]	0
14	14	52	2,102	2020-02-01	[NULL]	0

Views Report

- View 1: Track the rental information of each student.




```
CREATE VIEW Student.[Students rental information] AS
SELECT si.College, si.StudentID, si.FirstName, si.LastName,
       si.Gender, bi.BooksName, r.RentalDate, r.DueDate
FROM (
       (
       Student.StudentInformation si
       INNER JOIN Rental.Rental r
       ON si.StudentID = r.StudentID
       )
       INNER JOIN Rental.BookCopy bc
       ON r.CopyID = bc.CopyID
    )
INNER JOIN Book.BookInformation bi
ON bc.BooksID = bi.BooksID;

SELECT * FROM Student.[Students rental information ]
ORDER BY College, StudentID;
```

	College	StudentID	FirstName	LastName	Gender	BooksName	RentalDate	DueDate
1	Art	2,105	Beyoneta	Beth	Female	Database Systems-A Practical Approach to Design Implementation and Management	2020-01-06	2020-02-05
2	Art	2,105	Beyoneta	Beth	Female	No Longer Human	2020-01-06	2020-02-05
3	Business	2,106	Jill	Valentine	Femail	Database Systems-A Practical Approach to Design Implementation and Management	2020-01-06	2020-02-05
4	Business	2,106	Jill	Valentine	Femail	No Longer Human	2020-01-06	2020-02-05
5	Business	2,108	Laura	Croft	Female	Database Systems-A Practical Approach to Design Implementation and Management	2021-01-05	2021-02-04
6	CS	2,104	Franklin	Deo	Male	No Longer Human	2020-03-17	2020-04-16
7	Engineering	2,101	John	Wick	Male	Database Systems-A Practical Approach to Design Implementation and Management	2020-01-05	2020-02-04
8	Engineering	2,101	John	Wick	Male	Principles Of DataBase Management	2020-01-05	2020-02-04
9	Engineering	2,101	John	Wick	Male	Kingdom Come	2020-01-05	2020-02-04
10	Engineering	2,101	John	Wick	Male	Innovative Bridge Design Handbook	2020-01-05	2020-02-04
11	Engineering	2,102	Michael	Townly	Male	Innovative Bridge Design Handbook	2020-01-02	2020-02-01
12	Engineering	2,102	Michael	Townly	Male	The Killing Joke	2020-01-02	2020-02-01
13	Engineering	2,102	Michael	Townly	Male	Microsoft SQL Server 2016-A beginner Guide	2020-08-05	2020-09-04
14	Engineering	2,110	Sam	Bridges	Male	Kingdom Come	2020-03-15	2020-04-14

Views Report

- View 2: Track the rental information of computer application.
- View 3: Track the rental information of comic.
- View 4: Track the rental information of civil engineering.
- View 5: Track the rental information of management

```
CREATE VIEW Book.[Computer Application Book Rental Information] AS
SELECT bt.Information, bi.BooksName, si.FirstName, si.LastName, si.College
FROM (
        (
            (
            Book.BookType bt
            INNER JOIN Book.BookInformation bi
            ON bt.BooksTypeID = bi.BooksTypeID
            )
        INNER JOIN Rental.BookCopy bc
        ON bi.BooksID = bc.BooksID
        )
    INNER JOIN Rental.Rental r
    ON bc.CopyID = r.CopyID
    )
INNER JOIN Student.StudentInformation si
ON r.StudentID = si.StudentID
WHERE bt.BooksTypeID = 2;
```

	Information	BooksName	FirstName	LastName	College
1	Computer Application	Database Systems-A Practical Approach to Design Implementation and Management	Beyoneta	Beth	Art
2	Computer Application	Database Systems-A Practical Approach to Design Implementation and Management	Jill	Valentine	Business
3	Computer Application	Database Systems-A Practical Approach to Design Implementation and Management	Laura	Croft	Business
4	Computer Application	Microsoft SQL Server 2016-A beginner Guide	Michael	Townly	Engineering
5	Computer Application	Database Systems-A Practical Approach to Design Implementation and Management	John	Wick	Engineering
6	Computer Application	Principles Of DataBase Management	John	Wick	Engineering

Table-Level Constraint

- Prevent books from being checked out when availability is 0

```
CREATE FUNCTION Rental.CheckAvailability (@CpID int)
RETURNS SMALLINT
AS
BEGIN
    DECLARE @Count SMALLINT = 0;
    SELECT @Count = COUNT(bc.CopyID)
    FROM Rental.BookCopy bc
    WHERE bc.CopyID = @CpID
        AND bc.Availability = 0;
    RETURN @Count;
END;

ALTER TABLE Rental.Rental ADD CONSTRAINT NotAvailable CHECK (Rental.CheckAvailability(CopyID) = 0);
```

- Prevent renting when there is a previous record of books returned after the due date

```
CREATE FUNCTION Rental.CheckDuedate (@StID int)
RETURNS SMALLINT
AS
BEGIN
    DECLARE @Count SMALLINT = 0;
    SELECT @Count = COUNT(re.StudentID)
    FROM Rental.Render re
    WHERE re.StudentID = @StID
        AND re.OverDue > 0;
    RETURN @Count;
END;

ALTER TABLE Rental.Rental ADD CONSTRAINT BanOverdue CHECK (Rental.CheckDuedate(StudentID) = 0);
```

Table-Level Constraint

- Check the status of the student card to prevent students from borrowing books with an inactive student card

```sql
-- Ban if card is inactivated.
CREATE FUNCTION Rental.CheckCardStatus (@SID varchar(30))
RETURNS smallint
AS
BEGIN
   DECLARE @Count smallint=0;
   SELECT @Count = COUNT(StudentID)
          FROM (SELECT StudentID
        FROM Student.StudentInformation si
        JOIN Student.CardInformation ci
        ON si.CardNumber = ci.Cardnumber
        WHERE ci.CardStatus = 'inactivated') a
          WHERE StudentID = @SID;
   RETURN @Count;
END;

ALTER TABLE Rental.Rental ADD CONSTRAINT BanInactiveCard CHECK (Rental.CheckCardStatus(StudentID) = 0);

--Ban inactivited card
INSERT INTO Rental.Rental (RentalID,StudentID,CopyID,StaffID,RentalDate,DueDate)
VALUES (1010,2107,65,118,'2020-03-18','2020-04-17');

--Allow activite card
INSERT INTO Rental.Rental (RentalID,StudentID,CopyID,StaffID,RentalDate,DueDate)
VALUES (1010,2108,61,118,'2020-03-18','2020-04-17');
```

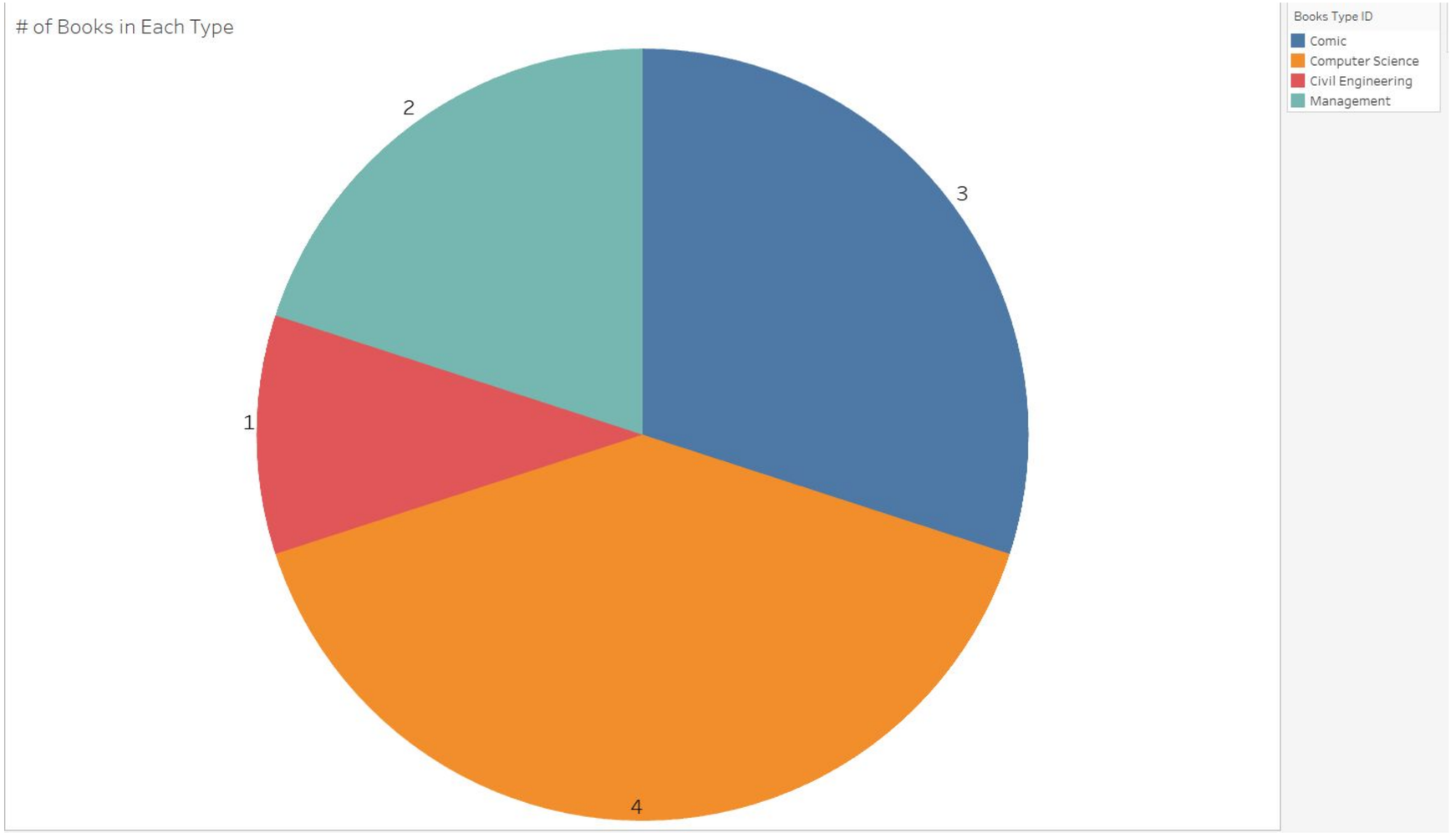
```
Started executing query at Line 26
Msg 547, Level 16, State 0, Line 2
The INSERT statement conflicted with the CHECK constraint "BanInactiveCard". The conflict occurred in database "Group6-Project", table "Rental.Rental", column 'StudentID'.
The statement has been terminated.
Total execution time: 00:00:00.089
```

Data Encryption

```
-- Create DMK
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Test_P@sswOrd';

-- Create certificate to protect symmetric key
CREATE CERTIFICATE TestCertificate
WITH SUBJECT = 'Card Information password',
EXPIRY_DATE = '2021-10-01';

-- Create symmetric key to encrypt data
CREATE SYMMETRIC KEY TestSymmetricKey
WITH ALGORITHM = AES_128
ENCRYPTION BY CERTIFICATE TestCertificate;

-- Open symmetric key
OPEN SYMMETRIC KEY TestSymmetricKey
DECRYPTION BY CERTIFICATE TestCertificate;


--upload new encrypt version of password
UPDATE Student.CardInformation
SET [Password] = EncryptByKey(Key_GUID(N'TestSymmetricKey'),convert(varbinary, 'zz1xzd'))
WHERE CardNumber = 12350;


SELECT * FROM Student.CardInformation;

SELECT CardNumber,convert(varchar,DECRYPTBYKEY([Password])) AS [Password],CardStatus
FROM Student.CardInformation;
```

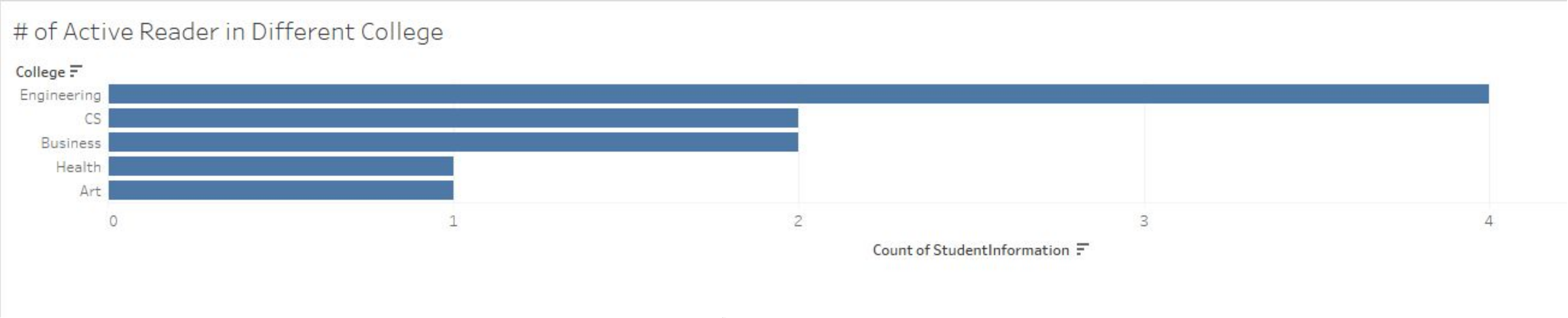
umber	ABC Password	ABC
12,341	§Kó¿íD§ y‹ZÌõy	p¶â/½òepÂ◆à*K9o¡—«gÃö‡9\(	Ac
12,342	§Kó¿íD§ y‹ZÌõy	Žô½³ä„pHáLüWLJ{u€éy◆mŠ6	Ac
12,343	§Kó¿íD§ y‹ZÌõy	ðfiæ"ÞXë¤ñî¸˜'P.§tWï"¦öp¸◆Õ	Ac
12,344	§Kó¿íD§ y‹ZÌõy	¯2.¸qÍU‡¢åCˆ	Ac
12,345	§Kó¿íD§ y‹ZÌõy	z;×¢òmÅöa«ê*‰69 i—óÖFGX›	Ac
12,346	§Kó¿íD§ y‹ZÌõy	Y¥9gÑ)$˜"ü‡Ö[[“4ZÝã7,d:³¡Â^	Ac
12,347	§Kó¿íD§ y‹ZÌõy	Ç◆mÉV4ËÂf™½;ëhˆ^†E°◆·w	Ac
12,348	§Kó¿íD§ y‹ZÌõy	|%çèsðb]¨ÃŒu r◆FËš×ªt◆6M	Ac
12,349	§Kó¿íD§ y‹ZÌõy	)ÒP»Àe@Í[ã«7€rèoô%Ãw§…Âú	Ac
12,350	§Kó¿íD§ y‹ZÌõy	íÛ)pØÜÂBÚghÄ¶í;€X¦Sd›z&‡f{¯	Ac

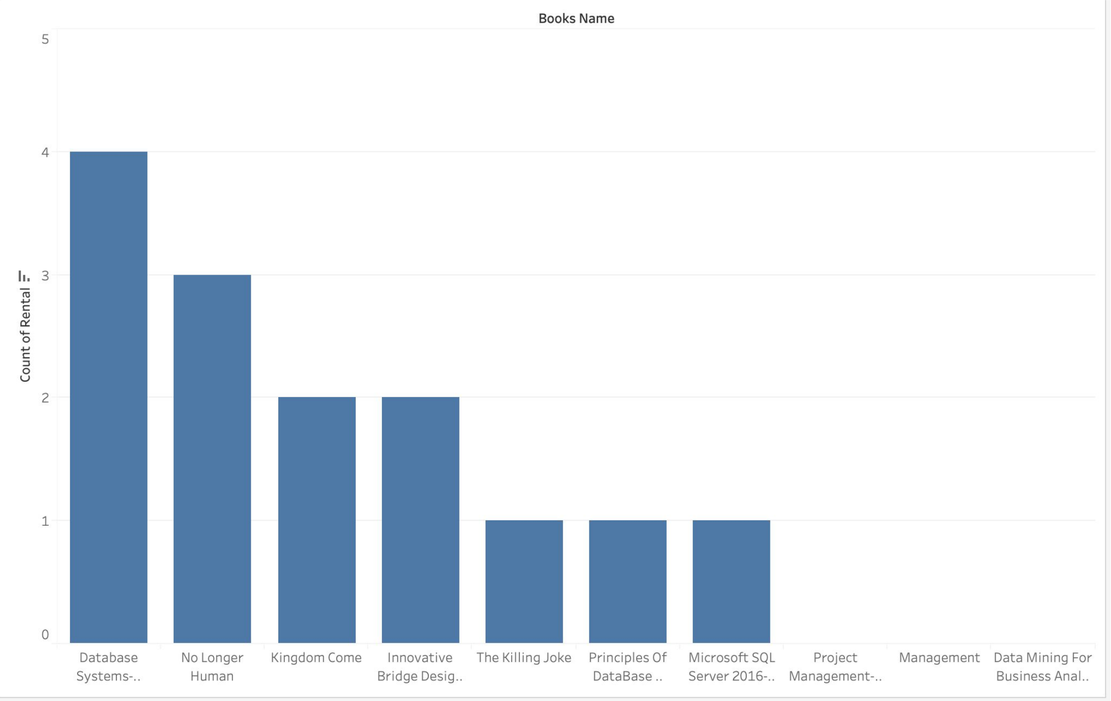
Reports & Visualizations



The maximum number of books is in Computer Science which is 4 in total and, Civil Engineering has the least number of books.

Reports & Visualizations





The College of Engineering has the most number of active readers however the most popular book of the library is *Database Systems-A Practical Approach to Design Implementation and Management*. The copies of this book is borrowed for 4 times.

INFO 6210 Data Management and Database Design

Library System

Thanks for Watching