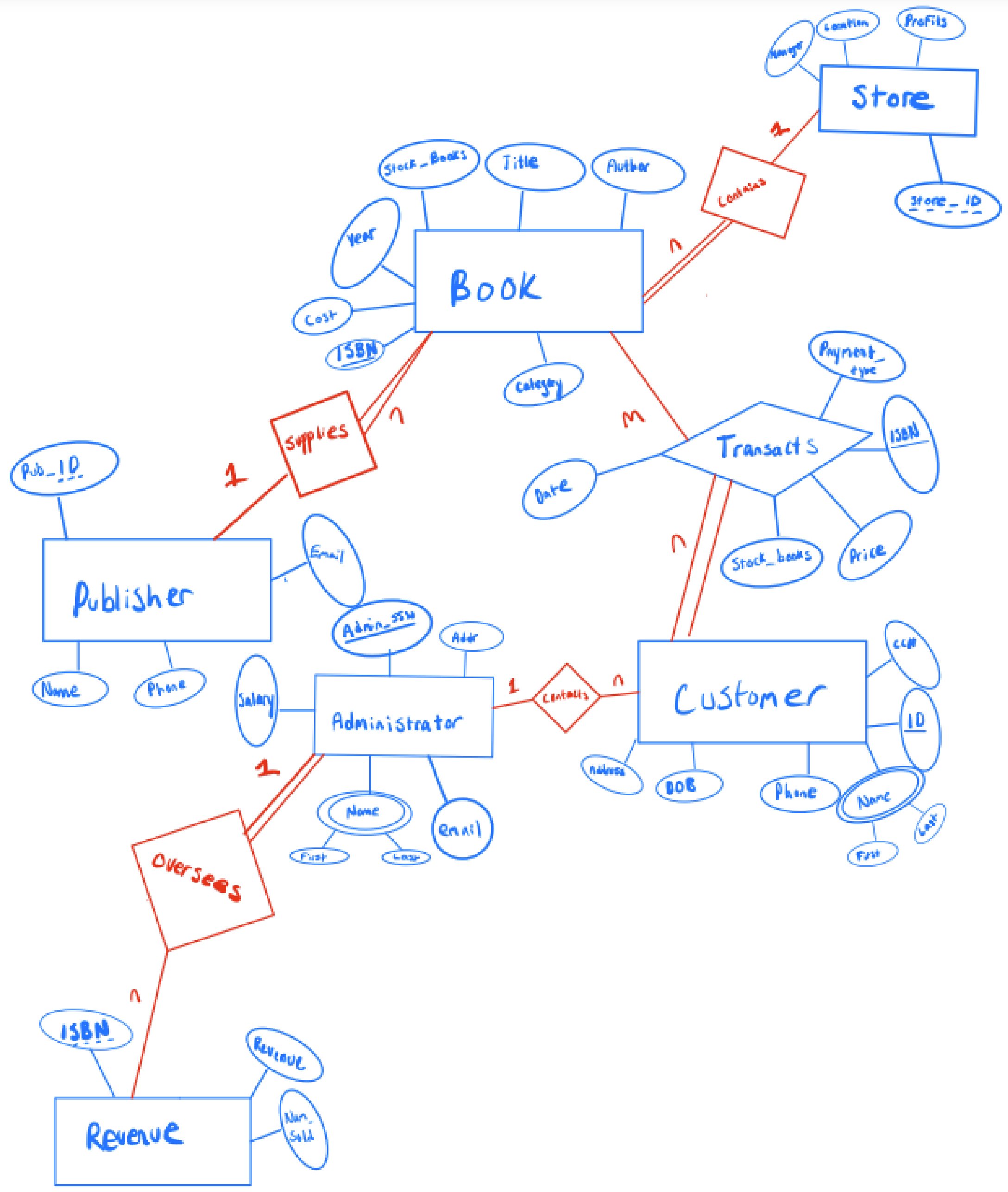
Final Project

Tamir Rastetter
Michael Frediani
Tyler Mante
Clark Hou

Table of Contents

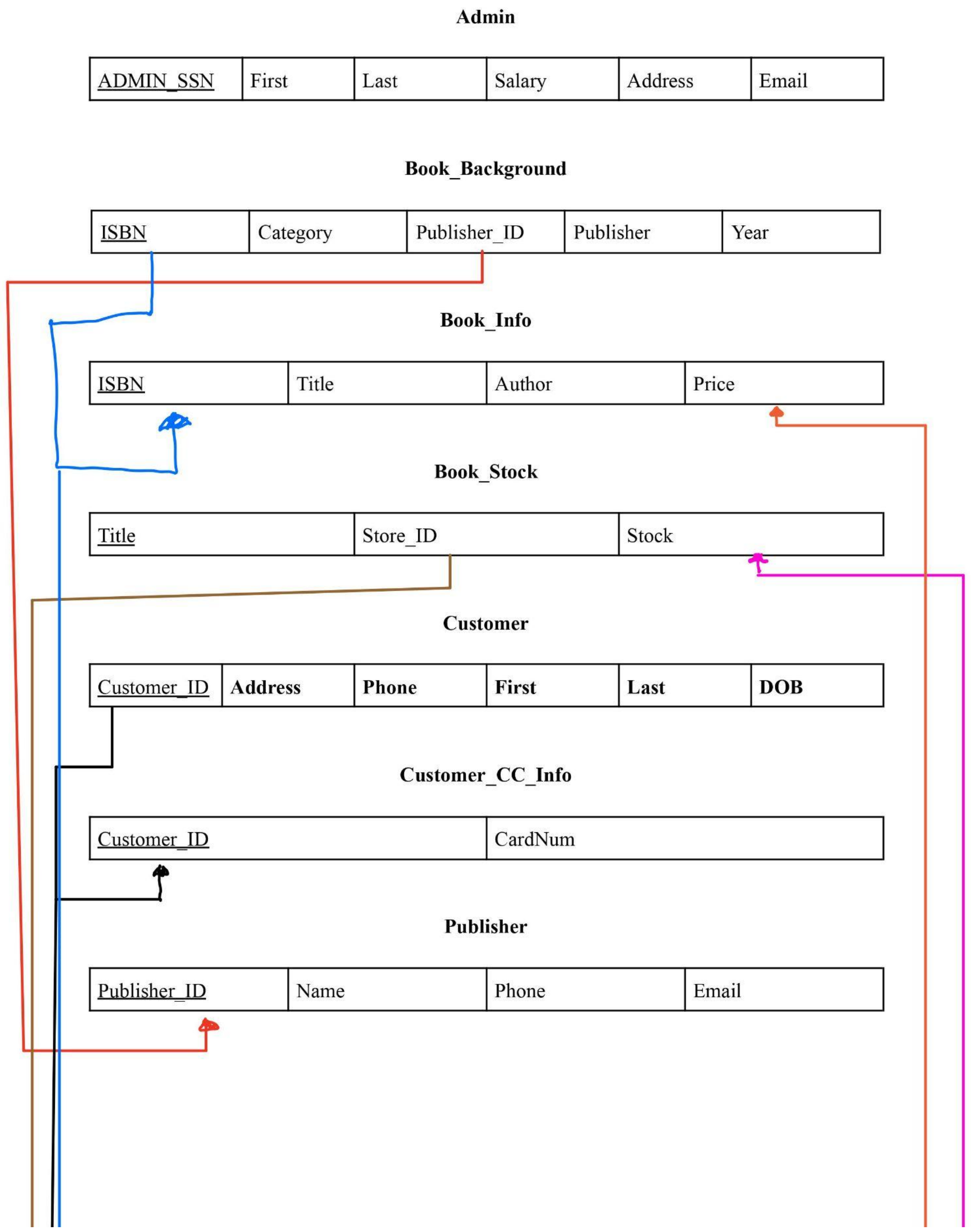
PART ONE-SECTION ONE
Database Description

ER-Model


Manager
Location
Profits
Store
Store_ID
1
Contains
n
Stock_Books
Title
Author
Year
Book
Cost
ISBN
Category
n
Supplies
1
m
Pub_ID
Payment_type
ISBN
Transacts
Date
Stock_books
Price
n
Publisher
Email
Admin_SSN
Addr
Name
Phone
Salary
Administrator
1
Contacts
n
Customer
CC#
ID
Address
DOB
Phone
Name
First
Last
1
Name
First
Last
email
Oversees
n
ISBN
Revenue
Revenue
Num_Sold

Relational Schema

This schema is after the normalization.



Admin

ADMIN_SSN	First	Last	Salary	Address	Email

Book_Background

ISBN	Category	Publisher_ID	Publisher	Year

Book_Info

ISBN	Title	Author	Price

Book_Stock

Title	Store_ID	Stock

Customer

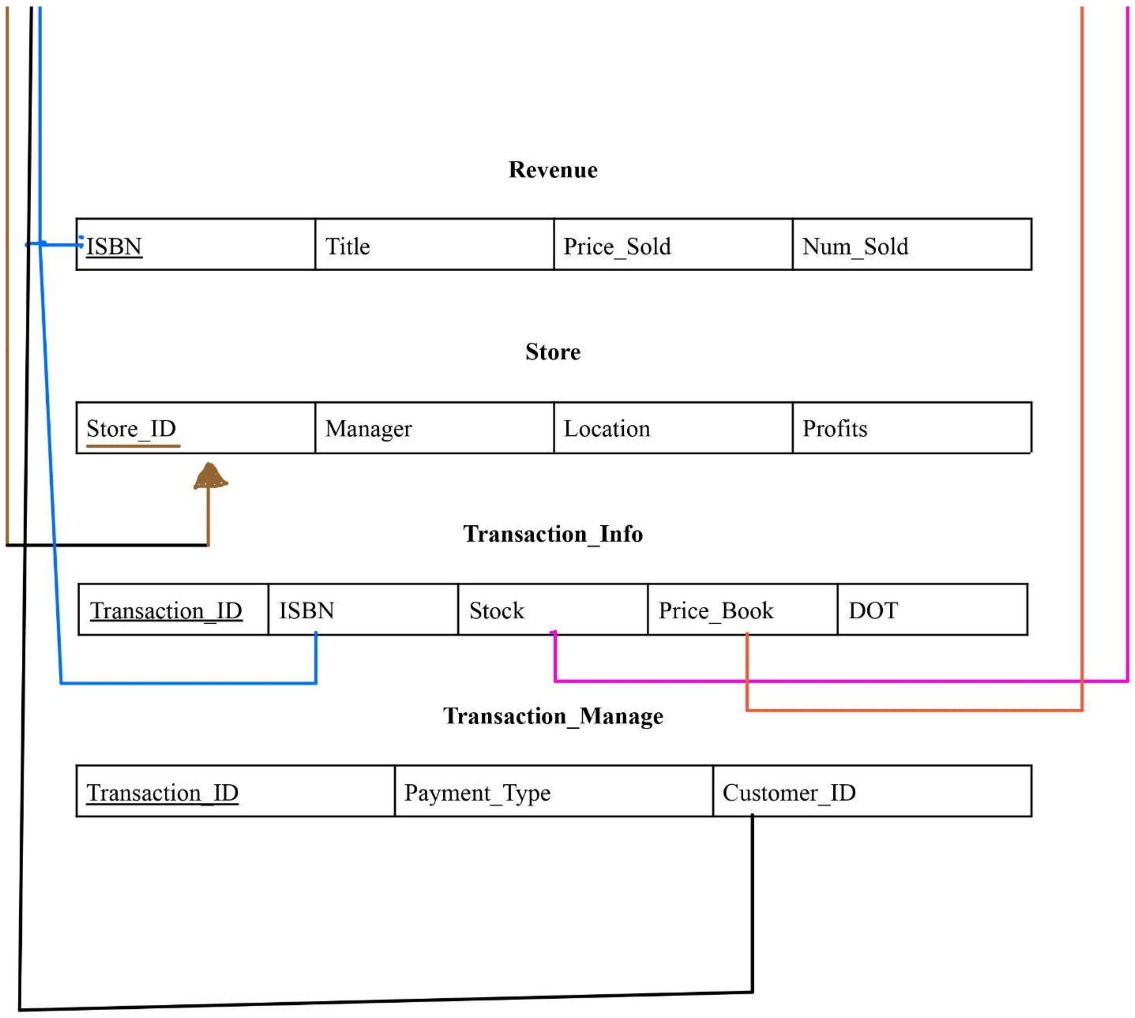
Customer_ID	**Address**	**Phone**	**First**	**Last**	**DOB**

Customer_CC_Info

Customer_ID	CardNum

Publisher

Publisher_ID	Name	Phone	Email



Revenue

ISBN	Title	Price_Sold	Num_Sold

Store

Store_ID	Manager	Location	Profits

Transaction_Info

Transaction_ID	ISBN	Stock	Price_Book	DOT

Transaction_Manage

Transaction_ID	Payment_Type	Customer_ID

Functional Dependencies

Book:

{ISBN} -> {PUBLISHER, CATEGORY, YEAR, TITLE, AUTHOR}
{TITLE, AUTHOR} -> {YEAR, PUBLISHER, CATEGORY}
*** assume title and author is a unique combination

Store:

{STORE_ID} -> {MANAGER, LOCATION}

Publisher:

{PUBLISHER_ID} -> {NAME, PHONE, EMAIL}
*** assume publishers could have same name, or change phones/emails

Customer:

{CUSTOMER_ID} -> {CC, ADDRESS, DOB, PHONE, FIRST, LAST}

Transacts:

{TRANSACT_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE, PAYMENT_TYPE, CUSTOMER_ID}

Administrator:

{ADMIN_SSN} -> {FIRST, LAST, SALARY, ADDRESS}

Revenue:

{ISBN} -> {REVENUE, NUM_SOLD}

Normalization

Book_Info:
{ISBN} -> {TITLE, AUTHOR, Price}

Book_Background:
{ISBN} -> {PUBLISHER_ID, PUBLISHER, YEAR, CATEGORY}

Book_Stock:
{TITLE, STORE-ID} -> {STOCK}

Store:
{STORE_ID} -> {MANAGER, LOCATION, PROFITS}

Publisher:
{PUBLISHER_ID} -> {NAME, PHONE, EMAIL}
*** assume publishers could have same name, or change phones/emails

Customer:
{CUSTOMER_ID} -> {ADDRESS, DOB, PHONE, FIRST, LAST}

Customer_CC_Info:
{CUSTOMER_ID} -> {CC}

Transacts_Info:
{TRANSACTION_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE}

Transacts_Manage
{TRANSACTION_ID} -> {PAYMENT_TYPE, CUSTOMER_ID}

Administrator:
{ADMIN_SSN} -> {FIRST, LAST, SALARY, ADDRESS}

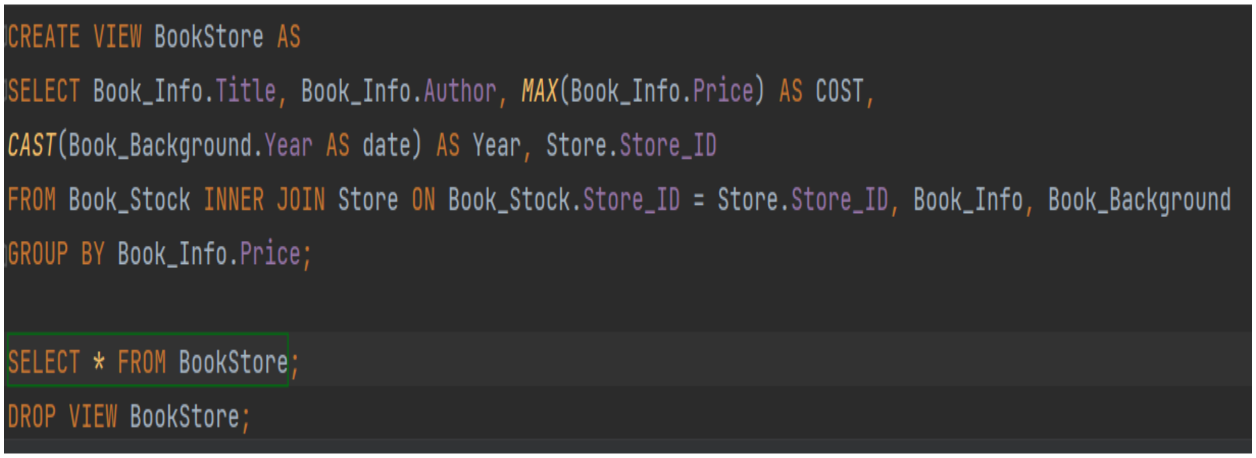
Revenue:
{ISBN} -> {REVENUE, NUM_SOLD}

Indexes

There were a plethora of indexes chosen to be implemented in this database. To begin with the Book relation, an index was introduced on the Author, Category, and Price attributes. An index on Author was introduced as to help out with searching on books titled by a specific author, many readers have a favorite author, and being able to quickly search said author can ensure no user is waiting an absorbent amount of time to see books by said author. Next, the category was an ideal candidate for an index as people enjoy reading a certain type of book, whether it be romance, murder mystery, non-fiction, and so on. Finally for the books, an index on Price was chosen as it greatly helps users search for books of a certain Price. This implementation allows users to quickly see which books are in their budget and purchase accordingly.

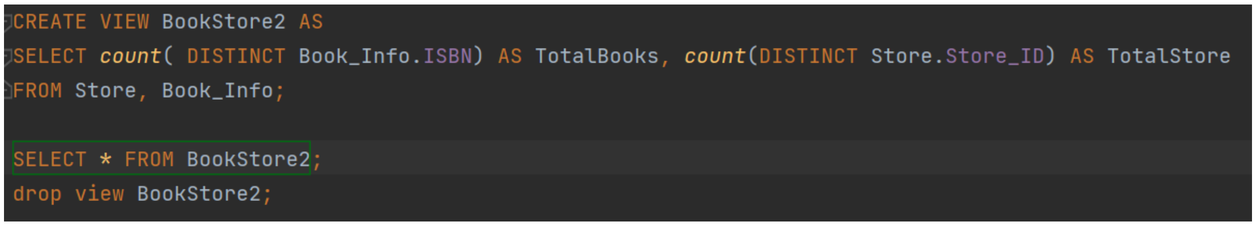
Views

View 1: Proposal for view to select book info and the store info it corresponds with as well as maximum Price from Book and Store Tables.



```
CREATE VIEW BookStore AS
SELECT Book_Info.Title, Book_Info.Author, MAX(Book_Info.Price) AS COST,
CAST(Book_Background.Year AS date) AS Year, Store.Store_ID
FROM Book_Stock INNER JOIN Store ON Book_Stock.Store_ID = Store.Store_ID, Book_Info, Book_Background
GROUP BY Book_Info.Price;

SELECT * FROM BookStore;
DROP VIEW BookStore;
```

View 2: Proposal for view to find total stores and total books from Book and Store Tables



```
CREATE VIEW BookStore2 AS
SELECT count( DISTINCT Book_Info.ISBN) AS TotalBooks, count(DISTINCT Store.Store_ID) AS TotalStore
FROM Store, Book_Info;

SELECT * FROM BookStore2;
drop view BookStore2;
```

Relational Algebra:

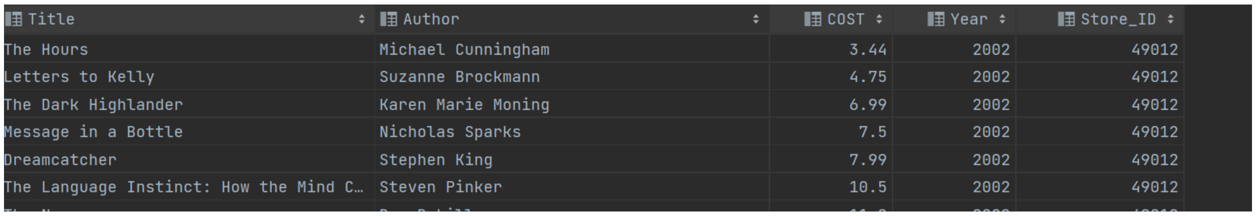
1)

$$BookStore(Isbn, title, author, Price, Year, Storeid) = \pi_{Isbn,title,author,Price,Year,Storeid}\left(Book \bowtie Store\right)$$

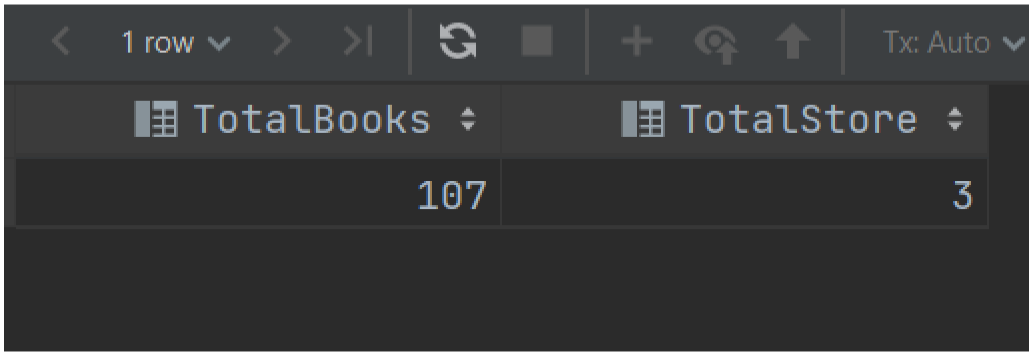
2) $$BookStore2\ (TotalStore, TotalBook) = \pi_{TotalStore,TotalBook}\ (Book \bowtie Store)$$

Example Output:

1)



Title	Author	COST	Year	Store_ID
The Hours	Michael Cunningham	3.44	2002	49012
Letters to Kelly	Suzanne Brockmann	4.75	2002	49012
The Dark Highlander	Karen Marie Moning	6.99	2002	49012
Message in a Bottle	Nicholas Sparks	7.5	2002	49012
Dreamcatcher	Stephen King	7.99	2002	49012
The Language Instinct: How the Mind C…	Steven Pinker	10.5	2002	49012

2) 

TotalBooks	TotalStore
107	3

Three Sample Transactions

1) Purchase a book that is the last one in Stock

```
UPDATE Transaction_Info
SET STOCK = STOCK - 1, Transaction_ID = 6784658790, ISBN = 9780060244194,
DOT = 11/18/2022
WHERE ISBN = 9780060244194;

UPDATE Transaction_Manage
SET Transaction_ID = 6784658790, Payment_Type = ‘Cash’, Customer_ID = 4002037
WHERE Customer_ID = 4002037;

UPDATE Revenue
SET Price_Sold = 19.99, Num_Sold = 1, ISBN = 6784658790, Title = ‘Done’
WHERE ISBN = 6784658790;

DELETE FROM Book_Info
WHERE ISBN = 9780060244194;
DELETE FROM Book_Background
WHERE ISBN = 9780060244194;
DELETE FROM Book_Stock
WHERE ISBN = 9780060244194;
```

2) Sample Return

```
Update Transaction_Info
SET Transaction_ID = 6784658790, ISBN = 9780060244194, STOCK = STOCK + 1,
DOT = 11/18/2022
WHERE ISBN = 9780060244194;

UPDATE Revenue
SET Price_Sold = -19.99, Num_Sold = 1, ISBN = 6784658790, Title = ‘Done’
WHERE ISBN = 6784658790;
```

3) A customer buys 2 books

```
UPDATE Transaction_Info
        SET STOCK = STOCK - 1, Transaction_ID = 6784658790, ISBN =
9780060244194, DOT = 11/18/2022
```

```
        WHERE ISBN = 9780060244194;

        UPDATE Transaction_Manage
        SET Transaction_ID = 6784658790, Payment_Type = ‘Cash’, Customer_ID =
4002037
        WHERE Customer_ID = 4002037;

        UPDATE Revenue
        SET Price_Sold = 19.99, Num_Sold = 1, ISBN = 6784658790, Title = ‘Done 1’
        WHERE ISBN = 6784658790;

UPDATE Transaction_Info
        SET STOCK = STOCK - 1, Transaction_ID = 1234567790, ISBN =
123321244194, DOT = 11/18/2022
        WHERE ISBN =123321244194;

        UPDATE Transaction_Manage
        SET Transaction_ID = 7906784658, Payment_Type = ‘Cash’, Customer_ID =
4002037
        WHERE Customer_ID = 4002037;

        UPDATE Revenue
        SET Price_Sold = 34.96, Num_Sold = 1, ISBN = 123321244194, Title = ‘Done
Two’
        WHERE ISBN = 123321244194;
```

PART ONE-SECTION TWO
User Manual

Sample SQL Queries

Simple Queries (WS2)

a. Find the titles of all books by Pratchett that Price less than $10

$\sigma_{Cost<10\ AND\ Author=\text{"Pratchett"}}(Book)$

```
SELECT TITLE
FROM Book_Info
WHERE Price < 10 AND AUTHOR Like “%Pratchett%”;
```

b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

$\pi_{Title,Date}\left(\left(\left(\sigma_{ID=\text{"Insert unique ID here"}}(Customer)\right) \bowtie_{ID=\text{Acct\_ID}} Transaction\right) \bowtie_{ISBN=ISBN} Book\right)$

```
Select Title, DOT From Book_Info, Transaction_Info, Transaction_Manage
Where Book_Info.ISBN = Transaction_Info.ISBN
And Transaction_Manage.Transaction_ID = Transaction_Info.Transaction_ID
And Transaction_Manage.Customer_ID = 4002037;
```

c. Find the titles and ISBNs for all books with less than 5 copies in stock

$\pi_{Title,ISBN}\left(\sigma_{\text{Stock\_Books}<5}(Book)\right)$

```
SELECT Book_Info.TITLE, Book_Info.ISBN
FROM Book_Stock, Book_Info
WHERE Book_Stock.STOCK < 5;
```

d. Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased

$\pi_{Name,Title}\left(\sigma_{Author=\text{"Pratchett"}}\left(\left(Customer \bowtie_{ID=\text{Acct\_ID}} Transaction\right) \bowtie_{ISBN=ISBN} Book\right)\right)$

```
Select First, Last, Title
From Book_Info, Transaction_Manage, Customer, Transaction_Info
Where Transaction_Manage.Customer_ID = Customer.Customer_ID
And Transaction_Manage.Transaction_ID = Transaction_Info.Transaction_ID
And Transaction_Info.ISBN = Book_Info.ISBN
And Author Like "%Pratchett%"
```

e. **Find the total number of books purchased by a single customer (you choose how to designate the customer)**

$$\mathfrak{F}_{COUNT\ ISBN}\left(\sigma_{ID=\text{Insert unique ID here}}(Customer)\right) \bowtie_{ID=\text{Acct\_ID}} Transaction$$

```
    SELECT COUNT(Transaction_Info.ISBN)
    FROM Customer, Transaction_Manage, Transaction_Info
    WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
      AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID
  AND Customer.Customer_ID = "insert unique ID here";
```

f. Find the customer who has purchased the most books and the total number of books they have purchased

$$\mathfrak{F}_{MAX}\ \mathfrak{F}_{COUNT\ ISBN}(Customer)\bowtie_{ID=\text{Acct\_ID}}Transaction(Customer)$$

```
SELECT COUNT(Transaction_Info.ISBN), First, Last
FROM Customer, Transaction_Manage, Transaction_Info
WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID

GROUP BY Customer.Customer_ID
ORDER BY 1 DESC
LIMIT 1;
```

3 Other Interesting Queries

a. Projecting the customer name of the customer who provided the most revenue

$$\pi_{First,Last}\left(Customer \bowtie_{ID=\text{Acct\_ID}} \left(\mathfrak{F}_{MAX\ \text{Price\_sold}}\left(Revenue \underset{ISBN=ISBN}{\bowtie} Transaction\right)\right)\right)$$

```
SELECT SUM(Transaction_Info.Price_Book), First, Last
FROM Customer, Transaction_Manage, Transaction_Info
WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID

GROUP BY Customer.Customer_ID
ORDER BY 1 DESC
LIMIT 1;
```

b. All of the titles of the book at the Columbus location

$$\pi_{Title}\left(\sigma_{\text{Book\_Stock.Store\_ID=Store.Store\_ID } AND\ Location=\text{'Columbus'}}(\text{Book\_Stock}, Store)\right)$$

SELECT Book_Stock.Title
FROM Book_Stock, Store
WHERE Book_Stock.Store_ID = Store.Store_ID AND Location = 'Columbus';

C. The publisher who supplied the most books to the company

Select Name, count(Name)
From Publisher, Book_Background
Where Book_Background.Publisher_ID = Publisher.Publisher_ID
Group By Name
Order By 2 DESC
Limit 1

Advanced Queries (WS3)

a. Provide a list of customer names, along with the total dollar amount each customer has spent.

SELECT SUM(Transaction_Info.Price_Book), First, Last
FROM Customer, Transaction_Manage, Transaction_Info
WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID

GROUP BY Customer.Customer_ID
ORDER BY 1 DESC;

b. Provide a list of customer names and phone numbers for customers who have spent more than the average customer.

```
SELECT Phone, First, Last
FROM Customer, Transaction_Info
GROUP BY First
HAVING SUM(Transaction_Info.Price_Book) >
   (

SELECT AVG(TotalSpent) AS AVGSPEND
FROM Transaction_Info,
   (SELECT SUM(Transaction_Info.Price_Book) AS TotalSpent, First, Last
               FROM Customer,
                   Transaction_Manage,
                   Transaction_Info
               WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
                 AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID
               GROUP BY Customer.Customer_ID))
```

c. Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.

```
SELECT Revenue.Title, Num_Sold
FROM Transaction_Info, Book_Stock, Revenue
WHERE Transaction_Info.Stock = Book_Stock.Stock AND Book_Stock.Title = Revenue.Title
GROUP BY Num_Sold
ORDER BY 2 DESC;
```

d. Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.

```
SELECT Revenue.Title, Price_Sold
FROM Transaction_Info, Book_Stock, Revenue
WHERE Transaction_Info.Stock = Book_Stock.Stock AND Book_Stock.Title = Revenue.Title
GROUP BY  Price_Sold
ORDER BY 2 DESC;
```

e. Find the most popular author in the database (i.e. the one who has sold the most books)

```
SELECT Author
FROM Transaction_Info, Book_Stock, Revenue, Book_Info
WHERE Transaction_Info.Stock = Book_Stock.Stock AND Book_Stock.Title = Revenue.Title
AND Book_Info.Price = Transaction_Info.Price_Book
ORDER BY 1 DESC
LIMIT 1;
```

f. Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)

```
SELECT Author
FROM Transaction_Info, Book_Stock, Revenue, Book_Info
WHERE Transaction_Info.Stock = Book_Stock.Stock AND Book_Stock.Title = Revenue.Title
AND Book_Info.Price = Transaction_Info.Price_Book
ORDER BY 1 DESC (
   SELECT *
   FROM Revenue
   GROUP BY Title
   HAVING Revenue.Price_Sold * Revenue.Num_Sold AS TOTAL REVENUE
)
```

g. Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

```
SELECT *
FROM Customer
   (SELECT Author
FROM Transaction_Info, Book_Stock, Revenue, Book_Info
WHERE Transaction_Info.Stock = Book_Stock.Stock AND Book_Stock.Title = Revenue.Title
AND Book_Info.Price = Transaction_Info.Price_Book
ORDER BY 1 DESC (
   SELECT *
   FROM Revenue
   GROUP BY Title
   HAVING Revenue.Price_Sold * Revenue.Num_Sold AS TOTAL REVENUE))
```

h. Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
SELECT AUTHOR
FROM BOOK_INFO
(
SELECT Phone, First, Last
FROM Customer, Transaction_Info
GROUP BY First
HAVING SUM(Transaction_Info.Price_Book) >
    (
SELECT AVG(TotalSpent) AS AVGSPEND
FROM Transaction_Info,
(SELECT SUM(Transaction_Info.Price_Book) AS TotalSpent, First, Last
            FROM Customer,
               Transaction_Manage,
               Transaction_Info
            WHERE Transaction_Manage.Customer_ID = Customer.Customer_ID
             AND Transaction_Info.Transaction_ID = Transaction_Manage.Transaction_ID
            GROUP BY Customer.Customer_ID)))
```

Table Descriptions

Admin

The Admin table corresponds to the administrator (this is the same as an employee). The attributes include SSN (INTEGER), Address (TEXT), Salary (Integer), First (TEXT), Last (TEXT), and Email (TEXT). The primary key is the SSN, this will uniquely identify the administrator that is being requested. The Address, Salary, First, Last and Email represent the address, salary, first and last name and email of the given administrator. Finally, none of these attributes are allowed to be NULL.

Book_Background

The Book_Background corresponds to the background of the book. This includes the ISBN (TEXT) (primary key), Category (TEXT), Publisher_ID (INTEGER), Publisher (TEXT), and Year (INTEGER) of the book. These attributes are rather self explanatory except for the Publisher_ID, this is a foreign key that helps map this relation to the publisher, the Publisher_ID also uniquely identifies the publisher. The ISBN and Publisher_ID can not be null as they are keys. Lastly, the ISBN can be used to connect this table to other tables that contain ISBN.

Book_Info

The Book_Info table corresponds to the information of the book. This includes the ISBN (TEXT), Title (TEXT), Author (TEXT), and Price (REAL) of the book. The ISBN is a primary key. The Title, Author, and Price of the book maps to the title, author, and price of the book. The ISBN can not be NULL and can be used to connect and map to the other tables regarding books.

Book_Stock

The Book_Stock table corresponds to the stock (amount) of the book present at the stores. The attributes include the Title (TEXT) (primary key), Store_ID (INTEGER) (foreign key), and Stock (INTEGER) of the book. The Title of the book is what uniquely identifies the book in this case (no book may have the same title). The Store_ID is the unique identifier for the book and helps see which location the book is present at. Additionally, the Stock attribute sees the total amount of books present at the store. The Title and Store_ID must not be NULL.

Store

The store table corresponds to the real world entity of a store. The attributes include: Store_ID (INTEGER) (primary key), Manager (TEXT), Location (TEXT), and Profits (TEXT).

The Store_ID is the unique identifier for the store, it is a sequence of 5 characters that identifies the store. The Manager, Location, and Profits are all rather self-explanatory, they are the manager of the store, the location of the store (city of store - we only have one store per city max), and the profits of the store (all-time). The only restraint is that the Store_ID is not null.

Publisher

The publisher table corresponds to the real world entity of a publisher. The attributes include: Publisher_ID (INTEGER), Name (TEXT), Phone (TEXT), Email (TEXT). The primary key is Publisher_ID, the assumption is that different publishers could have the same name, this allows us to uniquely identify each publisher. The other three attributes are rather simple, Name, Phone, Email all represent the name, phone and email of the publisher. All of these attributes could change as time goes on (company gets bought out, changes phone or email), while the Publisher_ID will stay the same. Lastly, the only restriction present is that the Publisher_ID must not be null.

Customer

The customer table corresponds to the real world entity of a customer. The attributes include Customer_ID (INTEGER), Address (TEXT), DOB (TEXT) (Date of Birth), Phone (TEXT), First (TEXT), and Last (TEXT). The primary key is ID, this is a 7 character in a unique sequence that can identify any customer. The Address, DOB, Phone, First and Last are all what they seem to be. The Address, DOB, Phone, First and Last is the address, date of birth, phone number, and first and last name of the customer.

Customer_CC_Info

This table contains the Customer_ID (INTEGER) (primary key) and the Credit Card number for the customer (attribute name CardNum) (REAL). This is separated from the Customer relation as to enforce normalization as well as enforce privacy and security concerns. Only restriction is that Customer_ID is not null.

Transaction_Info

The Transaction_Info table helps us keep basic data on a said transaction. The attributes include Transaction_ID (REAL) (primary key), ISBN (TEXT) (foreign key), Stock (INTEGER) (foreign key), Price_Book (REAL) (foreign key), DOT (REAL) (Date of Transaction). The Transaction_ID is how you can uniquely identify a transaction, the assumption present is that every time a book is purchased, a transaction ID is created for each book. This helps in situations where a customer purchases multiple books but only wishes to return one. Additionally, the

ISBN, Stock, and Price_Book are all foreign keys which map to other tables to further connect the database. Lastly, the DOT is the date of transaction which makes it easier to find the transaction for which is being searched. Every field must not be NULL.

Transaction_Manage

The Transaction_Manage table contains private information about the transaction. The attributes include Transaction_ID (REAL) (primary key), Payment_Type (TEXT), and Customer_ID (INTEGER). The Transaction_ID helps connect this table to the Info table. The payment_type is further information on how the book(s) were paid for, whether cash or credit card. Lastly, the Customer_ID is a foreign key that helps map to the specific customer that purchased items. None of these fields can be NULL.

Revenue

The revenue table corresponds to how much revenue a certain book generates. The attributes include ISBN (TEXT) (primary key), Title (TEXT), Price_Sold (REAL), and Num_Sold (INTEGER). The ISBN will be used to weakly identify the book, and the Num_Sold and Revenue will be used to see how many copies of the book sold and the total gross sum of revenue the book has generated. With the title being used to further identify the book. All of the attributes must not be null, this is our extra feature as it will allow us to gain insights into which books sell the most.

INSERT & DELETE Syntax

Book

Insert:

```
INSERT INTO Book_Info(ISBN, Title, AUTHOR, PRICE)
Values(9780060244195,"The Giving Tree", "Shel Silverstein", 20);

INSERT INTO Book_Background(PUBLISHER_ID, PUBLISHER, CATEGORY,
YEAR)
VALUES (63, "HarperCollins", "Childrens" , 1964);

INSERT INTO Book_Stock(TITLE, STORE_ID, STOCK)
VALUES( "The Giving Tree" , 30812, 19);

INSERT INTO Publisher(PUBLISHER_ID, NAME, PHONE, EMAIL)
VALUES(63, "HarperCollins", "330-410-8907", "HarperCollins”@gmail.com”");
```

Delete:

```
DELETE FROM Book_Info
WHERE ISBN = 9780060244194;
DELETE FROM Book_Background
WHERE ISBN = NULL;
DELETE FROM Book_Stock
WHERE ISBN = NULL;
```

Publisher

Insert:

```
INSERT INTO Publisher(PUBLISHER_ID, NAME, PHONE, EMAIL)
VALUES(63, Frediani Printing, 4127243309, fredianiprinting@gmail.com);
```

Delete:

```
DELETE FROM Publisher
WHERE PUBLISHER_ID = 63;
```

Author

Insert:

CREATE TABLE AUTHOR
(AUTHOR_ID CHAR(8)
FIRST VARCHAR(20)
LAST VARCHAR(20));
INSERT INTO Author(FIRST, LAST)
VALUES(12312323, ‘Colin’, ‘Cowherd’)

Delete:

DELETE FROM Author
WHERE Author_ID = 12312323)

Customer

Insert:

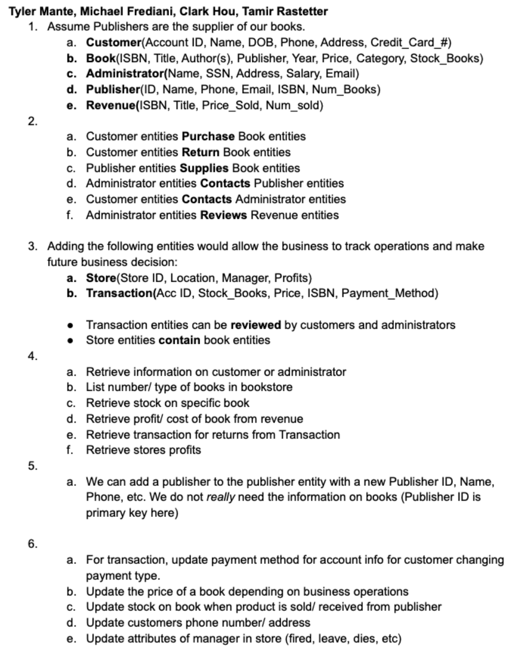
INSERT INTO Customer(CUSTOMER_ID, ADDRESS, DOB, PHONE, FIRST, LAST)
VALUES (8134091, ‘1667 Percheron Street’, “04-13-3004”, “412-724-1234”, “Noah”, “Blayney”)
INSERT INTO Customer_CC_INFO(CardNum)
VALUES(4430440090612386)

Delete:

DELETE FROM Customer
WHERE CUSTOMER_ID = 8134091

PART ONE-SECTION THREE
Graded Checkpoint Documents

Checkpoint One



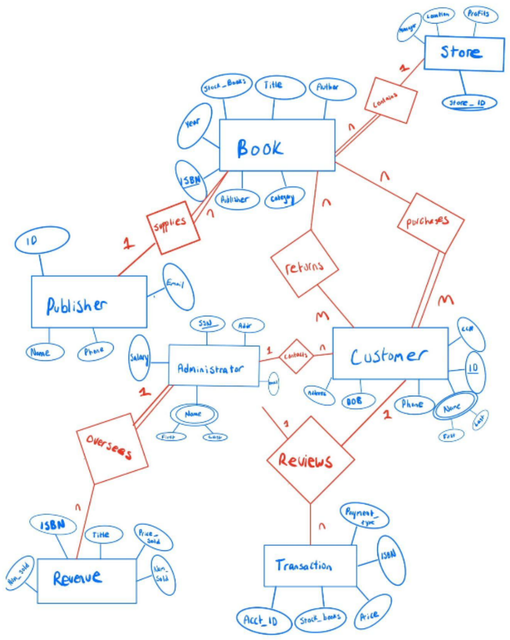
Tyler Mante, Michael Frediani, Clark Hou, Tamir Rastetter

1. Assume Publishers are the supplier of our books.
   a. **Customer**(Account ID, Name, DOB, Phone, Address, Credit_Card_#)
   b. Book(ISBN, Title, Author(s), Publisher, Year, Price, Category, Stock_Books)
   c. Administrator(Name, SSN, Address, Salary, Email)
   d. Publisher(ID, Name, Phone, Email, ISBN, Num_Books)
   e. Revenue(ISBN, Title, Price_Sold, Num_sold)
2.
   a. Customer entities **Purchase** Book entities
   b. Customer entities **Return** Book entities
   c. Publisher entities **Supplies** Book entities
   d. Administrator entities **Contacts** Publisher entities
   e. Customer entities **Contacts** Administrator entities
   f. Administrator entities **Reviews** Revenue entities

3. Adding the following entities would allow the business to track operations and make future business decision:
   a. Store(Store ID, Location, Manager, Profits)
   b. Transaction(Acc ID, Stock_Books, Price, ISBN, Payment_Method)

   - Transaction entities can be **reviewed** by customers and administrators
   - Store entities **contain** book entities
4.
   a. Retrieve information on customer or administrator
   b. List number/ type of books in bookstore
   c. Retrieve stock on specific book
   d. Retrieve profit/ cost of book from revenue
   e. Retrieve transaction for returns from Transaction
   f. Retrieve stores profits
5.
   a. We can add a publisher to the publisher entity with a new Publisher ID, Name, Phone, etc. We do not *really* need the information on books (Publisher ID is primary key here)

6.
   a. For transaction, update payment method for account info for customer changing payment type.
   b. Update the price of a book depending on business operations
   c. Update stock on book when product is sold/ received from publisher
   d. Update customers phone number/ address
   e. Update attributes of manager in store (fired, leave, dies, etc)



7.

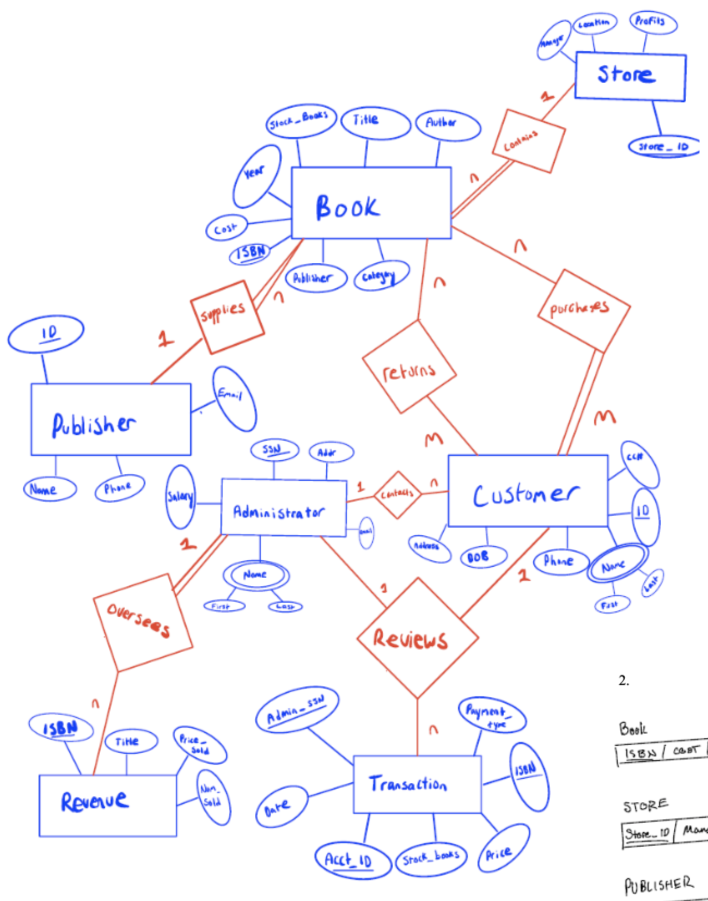
Feedback: "is there a relationship between TRANSACTION and BOOK. Because I think it is the purchases process generate the transaction. And also why not create a relationship between BOOK and REVENUE? At last, you mentioned that customers can return book, but I did not found corresponding thing in the ER-diagram."

REVISION: See ER-Model Page 4 for revisions, the final ER-Model that was deemed to need no revisions by the grader from Checkpoint Four feedback.

Checkpoint Two

Tyler Mante, Michael Frediani, Clark Hou, Tamir Rastetter

1.



3.

a.

$$\sigma_{Cost<10\ AND\ Author=\text{"Pratchett"}}(Book)$$

b.

$$\pi_{Title,Date}\left(\left(\left(\sigma_{ID=\text{"Insert unique ID here"}}(Customer)\right) \bowtie_{ID=\text{Acct\_ID}} Transaction\right) \bowtie_{ISBN=ISBN} Book\right)$$

c.

$$\pi_{Title,ISBN}\left(\sigma_{\text{Stock\_Books}<5}(Book)\right)$$

d.

$$\pi_{Name,Title}\left(\sigma_{Author=\text{"Pratchett"}}\left(\left(Customer \bowtie_{ID=\text{Acct\_ID}} Transaction\right) \bowtie_{ISBN=ISBN} Book\right)\right)$$

e.

$$\mathfrak{F}_{COUNT\ ISBN}\left(\sigma_{ID=\text{Insert unique ID here}}(Customer)\right) \bowtie_{ID=\text{Acct\_ID}} Transaction$$

f.

$$\mathfrak{F}_{MAX}\ \mathfrak{F}_{COUNT\ ISBN}{}_{(Customer)\bowtie_{ID=\text{Acct\_ID}}Transaction}(Customer)$$

4.

a. Projecting the customer name of the customer who provided the most revenue

$$\pi_{First,Last}\left(Customer \bowtie_{ID=\text{Acct\_ID}}\left(\mathfrak{F}_{MAX\ \text{Price\_sold}}\left(Revenue \underset{ISBN=ISBN}{\bowtie} Transaction\right)\right)\right)$$

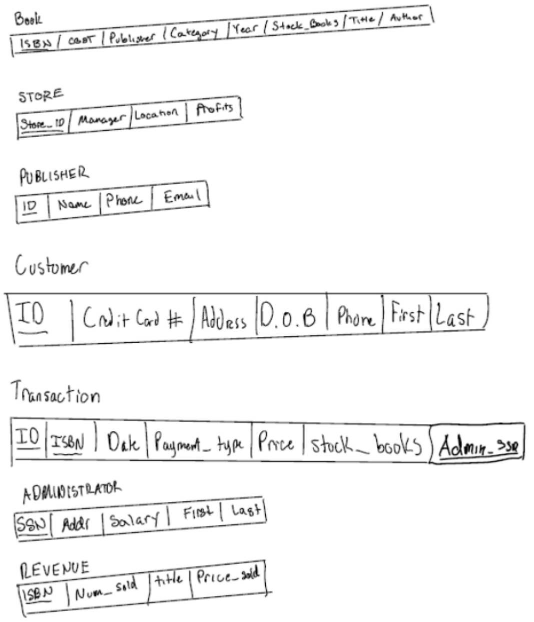
b. How many transactions has been admin-reviewed

$$\mathfrak{F}_{COUNT\ \sigma_{SSN=\text{"Insert unique SSN here"}}}\left(Transaction \bowtie_{\text{Admin\_SSN}=SSN} Administrator\right)$$

c. The publisher who supplied the most books

$$\pi_{Publisher}\left(\left(\mathfrak{F}_{MAX\ \text{Stock\_books}}\left(Book \underset{Publisher=Name}{\bowtie} Publisher\right)\right)\right)$$

2.



Book

ISBN	COST	Publisher	Category	Year	Stock_Books	Title	Author

STORE

Store_ID	Manager	Location	Profits

PUBLISHER

ID	Name	Phone	Email

Customer

ID	Credit Card #	Address	D.O.B	Phone	First	Last

Transaction

ID	ISBN	Date	Payment_type	Price	stock_books	Admin_SSN

ADMINISTRATOR

SSN	Addr	Salary	First	Last

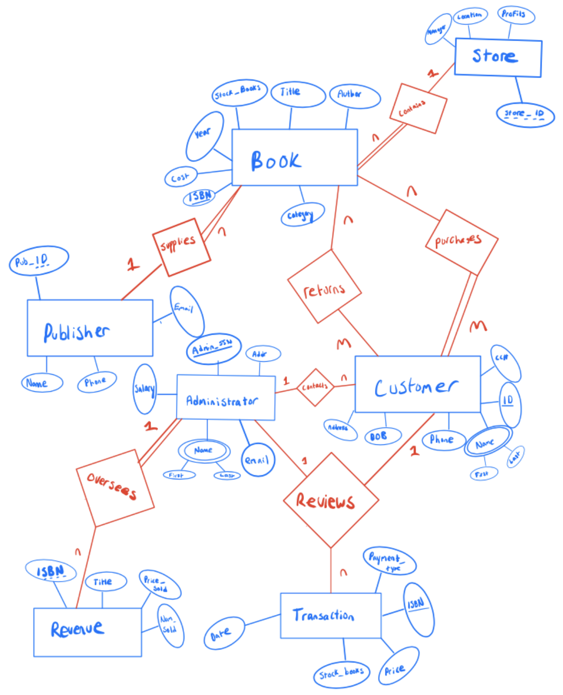
REVENUE

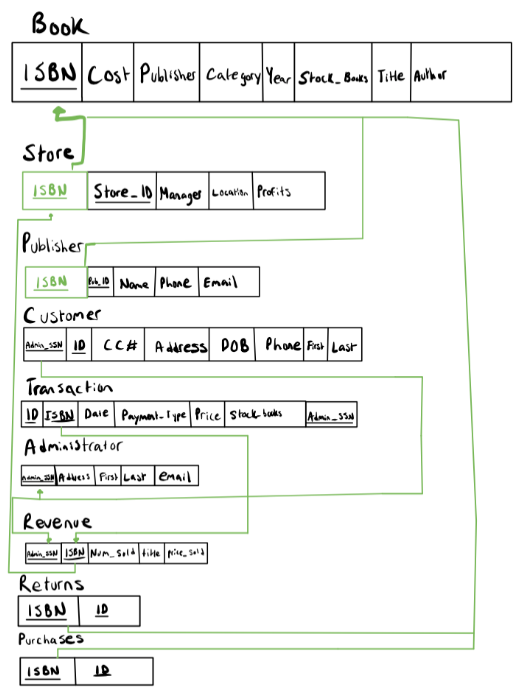
ISBN	Num_sold	title	Price_sold

Feedback: “you need to specify the foreign keys for the relational schema. There is an oversee relationship, but I could not find it in the relational schema. And the same case for the return and purchases relationships. I think it is better if we translate the ER-diagram into relational schemas rigorously following the rule. That can not only form a good schema but also help us to find if there are any unreasonable parts in the ER-diagram."

Revision: See Relational Schema Page 5, final model deemed to need no revision from the grader from Checkpoint Four feedback.

Checkpoint Three





```
CREATE SCHEMA BookProject;

CREATE TABLE Book
(
    ISBN CHAR(13) NOT NULL,
    COST INT      NOT NULL,
    PUBLISHER VARCHAR(20) NOT NULL,
    CATEGORY VARCHAR(20) NOT NULL,
    "YEAR" INT    NOT NULL,
    STOCK_BOOKS INT,
    TITLE VARCHAR(10000) NOT NULL,
    AUTHOR VARCHAR(10000) NOT NULL,
    PRIMARY KEY(ISBN)


);

CREATE TABLE Store
(
    ISBN CHAR(13) NOT NULL,
    STORE_ID VARCHAR(20) NOT NULL,
    MANAGER VARCHAR(30),
    LOCATION VARCHAR(20),
    PROFITS INT,
    PRIMARY KEY(STORE_ID),
    FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN)
);

CREATE TABLE Publisher
(
    ISBN CHAR(13) NOT NULL,
    PUBLISHER_ID CHAR(20) NOT NULL,
    "NAME" VARCHAR(25),
    PHONE INT NOT NULL DEFAULT 10,
    EMAIL VARCHAR(100) NOT NULL,

    PRIMARY KEY(PUBLISHER_ID),
    FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN)
);

CREATE TABLE Administrator
(
  SSN CHAR (9) NOT NULL,
  ADDRESS VARCHAR(50),
  FIRST_NAME VARCHAR(15) NOT NULL,
  LAST_NAME VARCHAR(15) NOT NULL,
  EMAIL VARCHAR(100) NOT NULL,


  PRIMARY KEY(SSN)


);
```

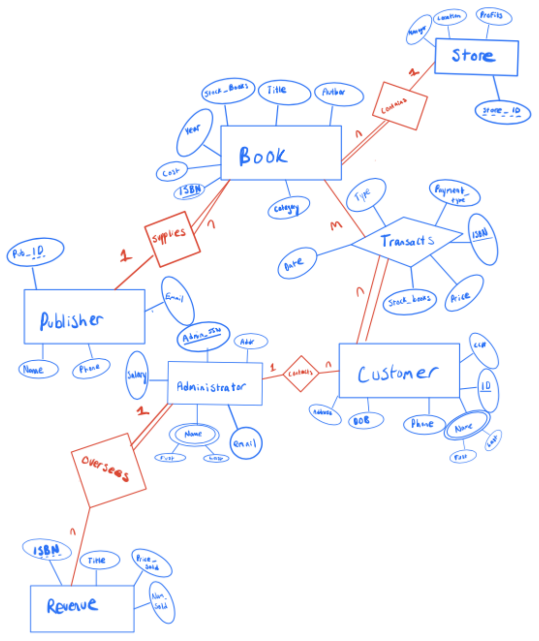
```
CREATE TABLE Customer
(
    ADMIN_SSN CHAR(9) NOT NULL,
    ID CHAR(7) NOT NULL,
    CC CHAR(16),
    ADDRESS (100),
    DOB DATE,
    PHONE CHAR(10),
    FIRST_NAME CHAR(15),
    LAST_NAME CHAR(20),
    PRIMARY KEY(ID),
    FOREIGN KEY (ADMIN_SSN) REFERENCES Administrator(SSN)
);

CREATE TABLE Revenue
(
  ADMIN_SSN CHAR(9) NOT NULL,
  ISBN CHAR(13) NOT NULL,
  NUM_SOLD INT,
  TITLE VARCHAR(10000) NOT NULL,
  PRICE_SOLD INT,

  FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
  FOREIGN KEY (ADMIN_SSN) REFERENCES Administrator(SSN)
);


CREATE TABLE Transaction
(
  TRANS_ID CHAR(7) NOT NULL,
  ISBN CHAR(13) NOT NULL,
  ADMIN_SSN CHAR(9) NOT NULL,
  "DATE" DATE NOT NULL,
  PAYMENT_TYPE VARCHAR(15) NOT NULL,
  PRICE INT NOT NULL,
  STOCK_BOOKS INT,

  PRIMARY KEY(TRANS_ID),
  FOREIGN KEY(ISBN) REFERENCES Revenue(ISBN),
  FOREIGN KEY (ADMIN_SSN) REFERENCES Administrator(SSN)
);

CREATE TABLE Returns
(
    ISBN CHAR(13) NOT NULL,
    Customer_ID CHAR(7) NOT NULL,

    FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(ID)
);

CREATE TABLE Purchases
(



  ISBN CHAR(13) NOT NULL,
  Customer_ID CHAR(7) NOT NULL,

  FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
  FOREIGN KEY (Customer_ID) REFERENCES Customer(ID)
);
```

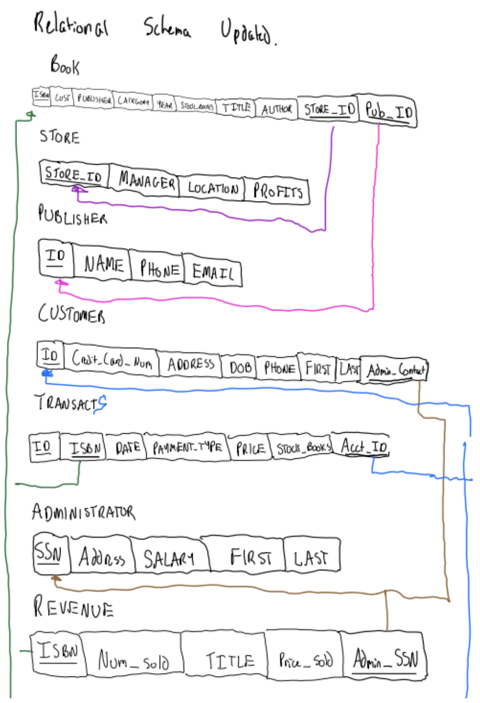
Checkpoint Three Cont.

Feedback: “Hi, I think I can't find your submission about the queries. I can't give you any appropriate feedback because of that.”

Revision: See Sample SQL Queries, Page 13-14 for missing Queries that were not included in the submission of this checkpoint.

Checkpoint Four





———————————- number 2

Number 2

L

Functional Dependencies:

Book:

{ISBN} -> {PUBLISHER, CATEGORY, YEAR, TITLE, AUTHOR}
{TITLE, AUTHOR} -> {YEAR, PUBLISHER, CATEGORY}
****assume title author is a unique combination

Store:

{STORE_ID} -> {MANAGER, LOCATION}

Publisher:

{PUBLISHER_ID} -> {NAME, PHONE, EMAIL}
{PHONE, EMAIL} -> {NAME}
****assume phone and email are both needed to uniquely identify name (we need them both)

Customer:

{CUSTOMER_ID} -> {CC#, ADDRESS, DOB, PHONE, FIRST, LAS

Transacts:

{TRANSACTION_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE, PAYMENT_TYPE, CUSTOMER_ID}

Administrator

{ADMIN_SSN} -> {FIRST, LAST, SALARY, ADDRESS}

Revenue:

{ISBN} - {NUM_SOLD, TITLE, PRICE_SOLD}

——————————

Number 3

Determine form make it 3NF

Book:

{ISBN} -> {PUBLISHER, CATEGORY, YEAR, TITLE, AUTHOR}
{TITLE, AUTHOR} -> {YEAR, PUBLISHER, CATEGORY}
****assume title author is a unique combination

This is not in 3NF, to make it into 3NF:
Seperate the two

Book:

{ISBN} -> {PUBLISHER, YEAR, CATEGORY}

Book_Info:

{ISBN} -> {TITLE, AUTHOR}

StorE:

Already in 3NF

{STORE_ID} -> {MANAGER, LOCATION}

**ASSUME MULTIPLE MANAGERS AT A SINGLE LOCATION
**ASSUME MULTIPLE LOCATIONS EXIST

Publisher:

{PUBLISHER_ID} -> {NAME, PHONE, EMAIL}
{PHONE, EMAIL} -> {NAME}
****assume phone and email are both needed to uniquely identify name (we need them both

This is NOT IN 3NF … we need:

Publisher_Name:

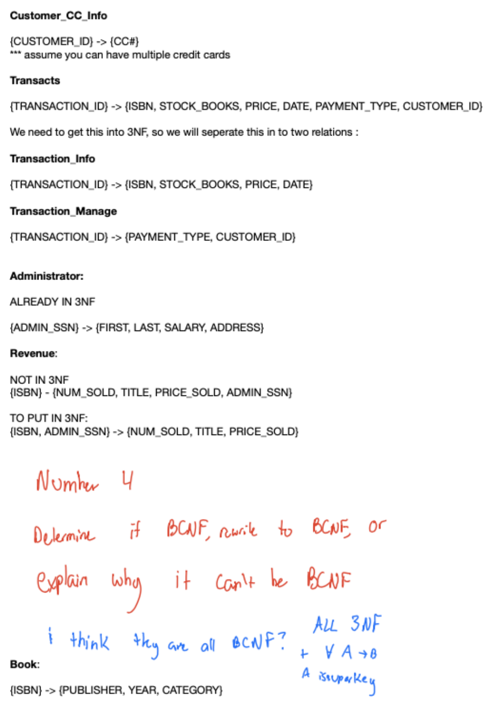
{PUBLISHER_ID} -> {NAME}

Publisher_Contact

{PUBLISHER_ID} -> {PHONE, EMAIL}

Customer:

{CUSTOMER_ID} -> {ADDRESS, DOB, FIRST, LAST}



Customer_CC_Info

{CUSTOMER_ID} -> {CC#}
*** assume you can have multiple credit cards

Transacts

{TRANSACTION_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE, PAYMENT_TYPE, CUSTOMER_ID}

We need to get this into 3NF, so we will seperate this in to two relations :

Transaction_Info

{TRANSACTION_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE}

Transaction_Manage

{TRANSACTION_ID} -> {PAYMENT_TYPE, CUSTOMER_ID}

Administrator:

ALREADY IN 3NF

{ADMIN_SSN} -> {FIRST, LAST, SALARY, ADDRESS}

Revenue:

NOT IN 3NF
{ISBN} - {NUM_SOLD, TITLE, PRICE_SOLD, ADMIN_SSN}

TO PUT IN 3NF:
{ISBN, ADMIN_SSN} -> {NUM_SOLD, TITLE, PRICE_SOLD}

Number 4

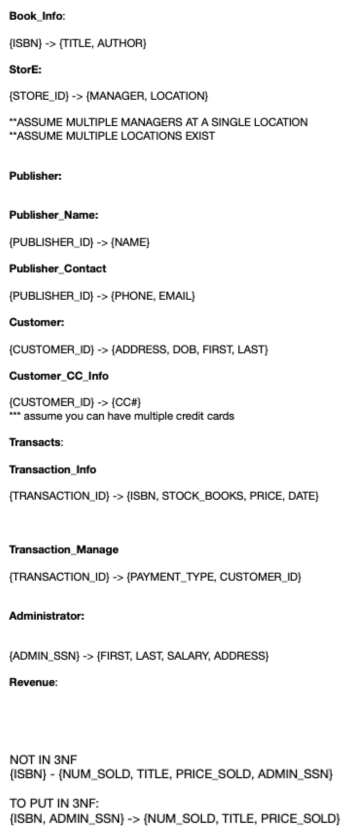
Determine if BCNF, rewrite to BCNF, or explain why it can't be BCNF

i think they are all BCNF?

ALL 3NF
+ $\forall A \to B$
A is superkey

Book:

{ISBN} -> {PUBLISHER, YEAR, CATEGORY}



Book_Info:

{ISBN} -> {TITLE, AUTHOR}

StorE:

{STORE_ID} -> {MANAGER, LOCATION}

**ASSUME MULTIPLE MANAGERS AT A SINGLE LOCATION
**ASSUME MULTIPLE LOCATIONS EXIST

Publisher:

Publisher_Name:

{PUBLISHER_ID} -> {NAME}

Publisher_Contact

{PUBLISHER_ID} -> {PHONE, EMAIL}

Customer:

{CUSTOMER_ID} -> {ADDRESS, DOB, FIRST, LAST}

Customer_CC_Info

{CUSTOMER_ID} -> {CC#}
*** assume you can have multiple credit cards

Transacts:

Transaction_Info

{TRANSACTION_ID} -> {ISBN, STOCK_BOOKS, PRICE, DATE}

Transaction_Manage

{TRANSACTION_ID} -> {PAYMENT_TYPE, CUSTOMER_ID}

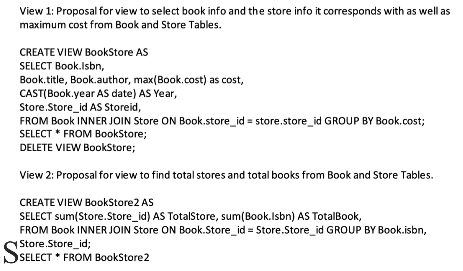
Administrator:

{ADMIN_SSN} -> {FIRST, LAST, SALARY, ADDRESS}

Revenue:

NOT IN 3NF
{ISBN} - {NUM_SOLD, TITLE, PRICE_SOLD, ADMIN_SSN}

TO PUT IN 3NF:
{ISBN, ADMIN_SSN} -> {NUM_SOLD, TITLE, PRICE_SOLD}



View 1: Proposal for view to select book info and the store info it corresponds with as well as maximum cost from Book and Store Tables.

```
CREATE VIEW BookStore AS
SELECT Book.Isbn,
Book.title, Book.author, max(Book.cost) as cost,
CAST(Book.year AS date) AS Year,
Store.Store_id AS Storeid,
FROM Book INNER JOIN Store ON Book.store_id = store.store_id GROUP BY Book.cost;
SELECT * FROM BookStore;
DELETE VIEW BookStore;
```

View 2: Proposal for view to find total stores and total books from Book and Store Tables.

```
CREATE VIEW BookStore2 AS
SELECT sum(Store.Store_id) AS TotalStore, sum(Book.Isbn) AS TotalBook,
FROM Book INNER JOIN Store ON Book.Store_id = Store.Store_id GROUP BY Book.isbn,
Store.Store_id;
SELECT * FROM BookStore2
```

SS

Feedback: “Hi, I think you did a good job! The only thing I'm thinking about is that if there is a FD as {Email} --> {Phone} for publisher. If this is your assumption that different publisher can have the same name, just ignore this.”

Revision: Our assumption is that different publishers have the same name, and as such no revisions are needed as stated in the feedback provided. We took no revisions from the feedback, but changed our database as see fit as we made the final database design.