2023

Database Project – Final Report

CSE 3241 – DATABASE SYSTEMS 1

QUENTIN SHARIER & LUCKY KATNENI

THE OHIO STATE UNIVERSITY

Final Report

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Contents

Database Description 2
ER Diagram 2
Relational Schema 3
Functional Dependencies 3
Level of Normalization 4
Indexes 5
Views 5
Views (Extra Credit) 6
User Manual 8
Table Explanations 8
Sample SQL Queries 11
Advanced Queries (Relational Algebra not Required) 13
Inserts 16
Deletes 17
Appendix 18
Graded Checkpoints 18
1. Checkpoint 1 18
2. Checkpoint 2 20
3. Checkpoint 3 22
4. Checkpoint 4 23

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Database Description

ER Diagram

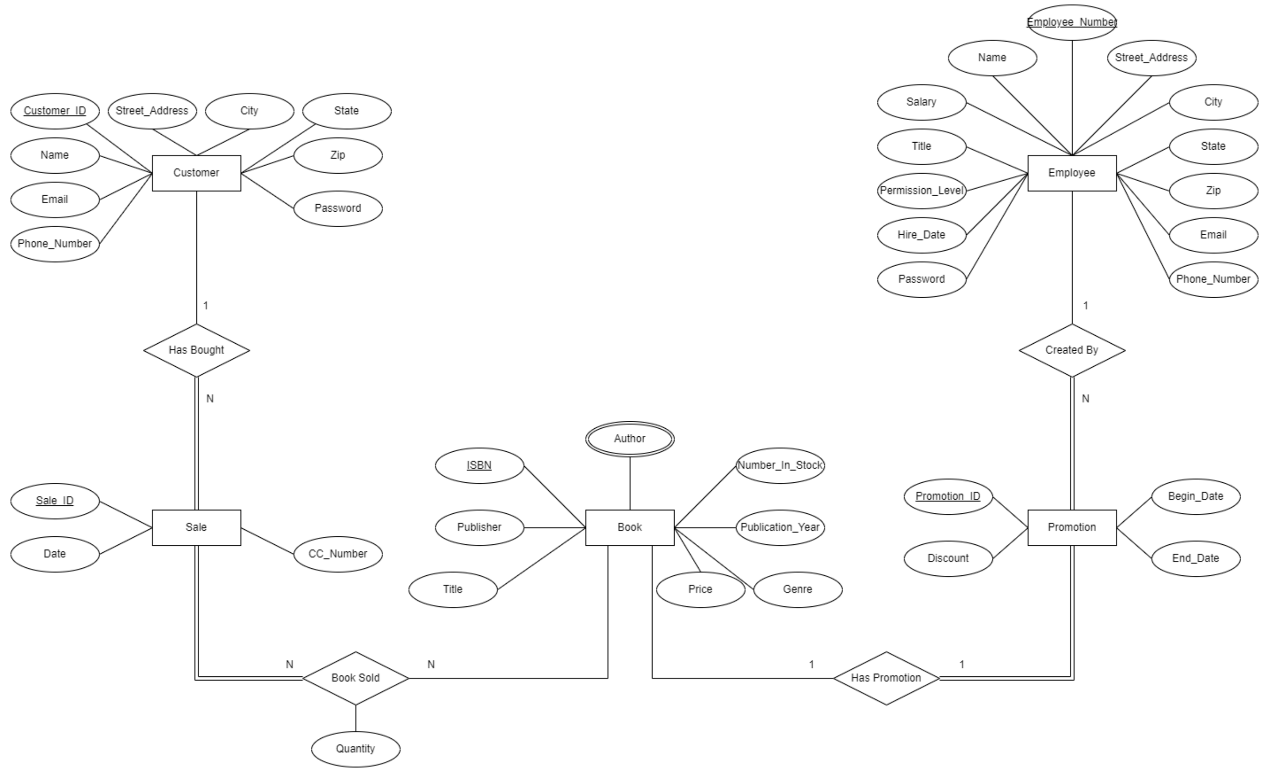
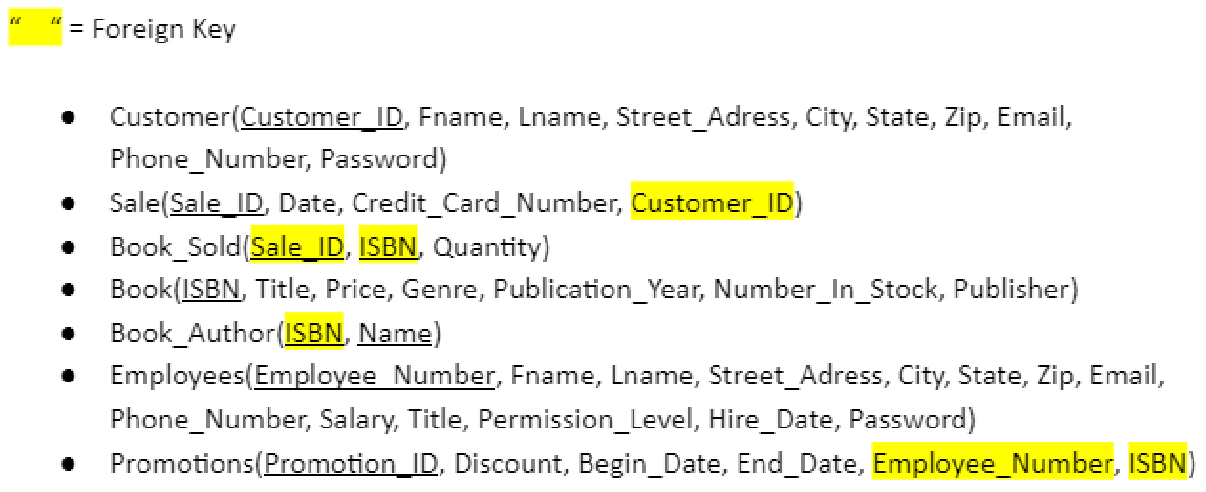


Figure 1 - ER Diagram

Relational Schema



" " = Foreign Key

- Customer(Customer_ID, Fname, Lname, Street_Adress, City, State, Zip, Email, Phone_Number, Password)
- Sale(Sale_ID, Date, Credit_Card_Number, Customer_ID)
- Book_Sold(Sale_ID, ISBN, Quantity)
- Book(ISBN, Title, Price, Genre, Publication_Year, Number_In_Stock, Publisher)
- Book_Author(ISBN, Name)
- Employees(Employee_Number, Fname, Lname, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password)
- Promotions(Promotion_ID, Discount, Begin_Date, End_Date, Employee_Number, ISBN)

Figure 2 - Relational Schema

Functional Dependencies

Customer

- {Customer_ID} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }
- {Email} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }
- {Phone_Number} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }

Sale

- {Sale_ID} -> { Sale_ID, Date, Credit_Card_Number, Customer_ID }

Book Sold

- {Sale_ID, ISBN} -> {Quantity}

Book

- {ISBN} -> { Title, Price, Genre, Publication_Year, Number_In_Stock, Publisher }

Book Author

- {ISBN} -> {Name}

Employees

- {Employee_Number} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }
- {Email} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }
- {Phone_Number} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }

Promotions

- {Promotion_ID} -> { Discount, Begin_Date, End Date, Employee_Number, ISBN }

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Level of Normalization

Customer

The Customer table is in 3NF because it is in 2NF, and no nonprime attributes are transitively dependent on the primary keys. Customer_ID is the primary key while Email and Phone_Number are both candidate keys, thus the transitivity rule does not apply to them as they are not non-prime.

Sale

The Sale table is in 3NF because it is in 2NF, and no nonprime attributes are transitively dependent on the primary keys. Credit_Card_Number creates no dependencies because many users can use the same credit card. Thus, the transitivity rule holds true.

Book Sold

The Book Sold table is in 3NF because it is in 2NF, and no non-prime attributes are transitively dependent on the primary keys.

Book

The Book table is in 3NF because it is in 2NF, and no non-prime attributes are transitively dependent on the primary keys.

Book Author

The Book Author table is in 3NF because it is in 2NF, and no non-prime attributes are transitively dependent on the primary keys.

Employees

The Employee table is in 3NF because it is in 2NF, and no nonprime attributes are transitively dependent on the primary keys. Employee_ID is the primary key while Email and Phone_Number are both candidate keys, thus the transitivity rule does not apply to them as they are not non-prime.

Promotions

The Promotions table is in 3NF because it is in 2NF, and no non-prime attributes are transitively dependent on the primary keys.

Indexes

1. CREATE INDEX Book_Genre_Idx ON Book (Genre);
   This index is created on the “Genre” column of the “Book” table. The purpose of this index is to improve the performance of queries that involve filtering or sorting by genre. The column “Genre” is likely to be used very frequently and there are many more books than there are employees or customers. By creating an index on the “Genre” column, the database can quickly locate the relevant rows and improve the overall performance of the query.

2. CREATE INDEX Book_Author_Idx ON Book_Author (Name);
   This index is created on the “Name” column of the “Book_Author” table. The purpose of this index is to improve the performance of queries that involve searching for books by a particular author’s name. This column is also more likely to be filtered through than most and there are many authors and many books. Overall, this improves the efficiency of queries that involve an author’s name.

3. CREATE INDEX Book_Sold_Idx ON Book_Sold (ISBN);
   This index is created on the “ISBN” column of the “Book_Sold” table. The purpose of this index is to improve the performance of queries that involve searching for sales data by a book’s ISBN. This could include finding the number of sales per book.

Views

1. Employee Promotions

   a. Description – The “employee promotions” view combines data from the “Employee” and “Promotions” tables to provide a summary of the number of promotions that each employee has been involved in.
   b. Relational Algebra - employee_promotions(Fname, Lname, Total_Promotions) $\leftarrow$ $\pi_{\text{Fname, Lname, Total\_Promotions}} \left(\left({}_{\text{Employee\_Number}}\mathcal{F}_{\text{COUNT(Promotion\_ID) AS Total\_Promotions}} \text{ Promotions } \right) \bowtie_{\text{Employee.Employee\_Number = Promotion.Employee\_Number}} \text{ Employee} \right)$
   c. SQL Creation:
```
CREATE VIEW employee_promotions AS
SELECT e.Fname, e.Lname, COUNT(p.Promotion_ID) AS Total_Promotions
FROM Employee e
LEFT JOIN Promotions p ON e.Employee_Number = p.Employee_Number
GROUP BY e.Fname, e.Lname;
```
   d. Sample:
```
Armando,Wilkins,3
Brianna,Dorsey,0
Darius,Sampson,1
Dustin,Compton,0
Elliott,Pearson,0
Herrod,Shields,1
```

Howard,Yates,0
Irma,Huber,2
Isabelle,Cantrell,0
Ivan,Simpson,1

2.

a. Description – The “book revenue” view provides the total revenue and number of books sold for each book in descending order of total revenue. It is derived from the “Book_Sold” and “Book” tables and calculates the revenue by multiplying the number of books sold by the book price.

b. Relational Algebra - book_revenue(ISBN, Title, Total_Sold, Total_Revenue) $\leftarrow$ ${}_{\text{ISBN, Title}}\mathcal{F}_{\text{SUM(Quantity) AS Total\_Sold, SUM(Quantity * Price) AS Total\_Revenue}}$ (Book_Sold $\bowtie_{\text{ISBN = Book.ISBN}}$ Book)

c. SQL Creation:

```
CREATE VIEW book_revenue AS
SELECT bs.ISBN, b.Title, SUM(bs.Quantity) AS Total_Sold, SUM(bs.Quantity *
b.Price) AS Total_Revenue
FROM Book_Sold bs
JOIN Book b ON bs.ISBN = b.ISBN
GROUP BY bs.ISBN
ORDER BY Total_Revenue DESC;
```

d. Sample:

```
782140114,Access 2002 Developer's Handbo,10,699.9
324113641,Introductory Econometrics: A M,6,647.7
130998516,Real World FPGA Design with Ve,2,166
471391123,Beyond Coso : Internal Control,3,150
262232197,Econometric Analysis of Cross ,2,149.9
743206061,The Second Time Around,4,104
471200247,The Data Warehouse Toolkit: Th,2,100
262024829,Financial Modeling ,1,70
375703764,House of Leaves,3,59.85
743467523,Dreamcatcher,6,47.94
```

Views (Extra Credit)

1.

a. Description – The “Top_Customers” view provides the total number of purchases each customer has provided, ordered in descending order so that one can see what customers make the most purchases.

b. Relational Algebra – Top_Customers(Fname, Lname, Total_Puchases) $\leftarrow$ ${}_{\text{Customer\_ID, Fname, Lname}}\mathcal{F}_{\text{COUNT(*) AS Total\_Purchases}}$ (Customer $\bowtie_{\text{Customer\_ID = Customer\_ID}}$ Sale)

c. SQL Creation:

```
CREATE VIEW Top_Customers AS
SELECT c.Fname, c.Lname, c.Customer_ID, COUNT(*) AS Total_Purchases
FROM Customer c
JOIN Sale s ON c.Customer_ID = s.Customer_ID
GROUP BY c.Customer_ID
ORDER BY Total_Purchases DESC;
```

d. Sample:

```
Ulla,Logan,18,5
Aretha,Erickson,19,4
Irma,Garza,11,4
Denton,Wells,14,3
Fritz,Snyder,13,3
Asher,Porter,9,3
Ronan,Shaw,7,3
Whitney,Marsh,6,3
Joel,Ingram,12,2
Ira,Mathews,10,2
```

2.

a. Description - The "Active Promotions" view provides a way for users to show the current active promotions and their respective books.

b. Relational Algebra – Active_Promotions(Promotion_ID, Title, Discount, Begin_Date, End_Date) ← $\pi_{\text{Promotion\_ID, Title, Discount, Begin\_Date, End\_Date}}$ ($\sigma_{\text{End\_Date} >= 20230426 \text{ AND Begin\_Date} <= 20230426}$(Promotions) $\bowtie_{\text{ISBN = ISBN}}$(Book))

c. SQL Creation:

```
CREATE VIEW Active_Promotions AS
SELECT p.Promotion_ID, b.Title, p.Discount, p.Begin_Date, p.End_Date
FROM Promotions p
JOIN Book b ON p.ISBN = b.ISBN
WHERE p.End_Date >= 20230426 AND p.Begin_Date <= 20230426;
```

d. Sample:

```
1,OCP: Oracle9i Certification Ki,0.51,20211224,20231101
5,The Rescuer: The O'Malley Seri,0.86,20210110,20240327
11,How the Mind Works,0.75,20230413,20230520
12,The Language Instinct: How the,0.37,20230125,20231023
13,Econometric Analysis of Cross ,0.18,20210804,20230616
14,Introductory Econometrics: A M,0.35,20210520,20230724
17,Chesapeake Blue,0.44,20230308,20230624
19,The Color of Magic,0.48,20210328,20230626
```

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

User Manual

Table Explanations

1. **Book:**
   This table represents a collection of books that the bookstore currently sells or has sold. Constraints have been built-in to ensure that ISBN is unique and that Price and Publication_Year are not nullable.
   Book Attributes:
   - ISBN (varchar(20)): The unique identifier for each book. It is not nullable and has been set as the primary key of the table.
   - Title (varchar(30)): The title of the book. It is not nullable.
   - Price (REAL): The price of the book. It is not nullable.
   - Genre (varchar(30)): The genre of the book.
   - Publication_Year (integer): The year in which the book was published. It is not nullable.
   - Publisher (varchar(40)): The name of the publisher for the book.

2. **Book_Author**
   This table represents the many-to-one relationship between books and their authors. It associates a book's ISBN with the name of one of its authors.
   Book_Author Attributes:
   - ISBN (varchar(20)): A foreign key that references the "ISBN" primary key in the "Book" table, of type integer. It has a constraint that, on update, the change will cascade to any referencing rows and, on delete, the value will be set to null for any referencing rows.
   - Name varchar((40)): The name of an author associated with a book.

3. **Book_Sold**
   This table represents the many-to-many relationship between a sale and the books that were sold during that transaction. It associates a Sale_ID with an ISBN and the number of books sold for that transaction.
   Book_Sold Attributes:
   - Sale_ID (integer): A foreign key that references the "Sale_ID" primary key in the "Sale" table. It has a constraint that, on update, the change will cascade to any referencing rows and, on delete, the value will be set to null for any referencing rows.
   - ISBN (varchar(30)): A foreign key that references the "ISBN" primary key in the "Book" table. It has a constraint that, on update, the change will cascade to any referencing rows and, on delete, the value will be set to null for any referencing rows.
   - Quantity (integer): The number of books sold in the transaction.

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

4. **Customer**
   This table represents the customers of the bookstore. Constraints have been built-in to ensure that Customer_ID, Fname, Lname, Email, and Password are not nullable and that Customer_ID is unique.
   Customer Attributes:
   - Customer_ID (integer): The unique identifier for each customer. It has been set as the primary key of the table and has a constraint to ensure it is not nullable and that it is unique.
   - Fname (varchar(15)): The first name of the customer. It is not nullable.
   - Lname (varchar(15)): The last name of the customer. It is not nullable.
   - Street_Address (varchar(30)): The street address of the customer.
   - City (varchar(20)): The city of the customer.
   - State (varchar(15)): The state of the customer.
   - Zip (integer): The zip code of the customer.
   - Email (varchar(80)): The email address of the customer. It is not nullable and has been set as unique.
   - Phone_Number (integer): The phone number of the customer.
   - Password (varchar(30)): The password of the customer. It is not nullable.

5. **Employee**
   This table represents the employees of the bookstore. Constraints have been built-in to ensure that Employee_Number, Fname, Lname, Email, Salary, Title, Hire_Date, and Password are not nullable and that Employee_Number is unique.
   Employee Attributes:
   - Employee_Number (integer): The unique identifier for each employee. It has been set as the primary key of the table and has a constraint to ensure it is not nullable and that it is unique.
   - Fname (varchar(15)): The first name of the employee. It is not nullable.
   - Lname (varchar(15)): The last name of the employee. It is not nullable.
   - Street_Address (varchar(40)): The street address of the employee.
   - City (varchar(30)): The city of the employee.
   - State (varchar(15)): The state of the employee.
   - Zip (integer): The zip code of the employee.
   - Email (varchar(40)): The email address of the employee. It is not nullable and has been set as unique.
   - Phone_Number (integer): The phone number of the employee.
   - Salary (integer): The salary of the employee. It is not nullable.
   - Title (varchar(30)): The job title of the employee. It is not nullable.
   - Hire_Date (integer): The date the employee was hired. It is not nullable.
   - Password (varchar(30)): The password of the employee. It is not nullable.

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

6. **Promotions**

   This table represents the promotions that are available for the bookstore's books. It associates a book's ISBN with a promotion ID and an employee number. Constraints have been built-in to ensure that Promotion_ID is unique and that ISBN, Discount, Begin_Date, End_Date, and Employee_Number are not nullable.

   Promotions Attributes:

   - Promotion_ID (integer): The unique identifier for each promotion. It is not nullable and has been set as the primary key of the table.
   - ISBN (varchar(30)): A foreign key that references the "ISBN" primary key in the "Book" table, of type integer. It has a constraint that, on update, the change will cascade to any referencing rows and, on delete, the value will be set to null for any referencing rows.
   - Discount (REAL): The discount percentage of the promotion. It is not nullable.
   - Begin_Date (integer): The date on which the promotion starts. It is not nullable.
   - End_Date (integer): The date on which the promotion ends. It is not nullable.
   - Employee_Number (integer): A foreign key that references the "Employee_Number" primary key in the "Employee" table, of type integer. It has a constraint that, on update, the change will cascade to any referencing rows and, on delete, the value will be set to null for any referencing rows.

7. **Sale**

   This table represents a sale made by the bookstore. It associates a sale ID with the date of the sale, the credit card number used to make the purchase, and the customer who made the purchase. Constraints have been built-in to ensure that Sale_ID is unique and that Date, Credit_Card_Number, and Customer_ID are not nullable.

   Sale Attributes:

   - Sale_ID (integer): The unique identifier for each sale. It is not nullable and has been set as the primary key of the table.
   - Date (integer): The date on which the sale was made. It is not nullable.
   - Credit_Card_Number (varchar(30)): The credit card number used to make the purchase. It is not nullable.
   - Customer_ID (integer): A foreign key that references the "Customer_ID"

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Sample SQL Queries

1. This query finds all books that are by the author Pratchett that cost less than $10.
   a. $\pi_{\text{Title}}(\sigma_{\text{Author='Pratchett'} \wedge \text{Price} < 10}(\text{Book}))$
   b.
```
SELECT title FROM
        Book NATURAL JOIN Book_Author WHERE
                        Book_Author.Name LIKE "%Pratchett%"
                        AND Price < 10;
```

2. This query gives all the titles and their dates of purchase made by a single customer.
   a. $\pi_{\text{Title, Date}}(\text{Book} * (\sigma_{\text{Customer\_ID} = 453}\text{Sale}))$
   b.
```
SELECT Title, Date FROM

        Book_Sold bs JOIN Sale s on s.Sale_ID = bs.Sale_ID

            JOIN Book B on B.ISBN = bs.ISBN WHERE
                        Customer_ID = 4;
```

3. This query finds the titles and ISBN's for all books with less than 5 copies in stock.
   a. $\pi_{\text{Title, ISBN}}(\sigma_{\text{Number\_In\_Stock} < 5}(\text{Book}))$
   b.
```
SELECT Title, ISBN
FROM Book
WHERE Number_In_Stock < 5;
```

4. This query gives all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased.
   a. $\pi_{\text{Name, Title}}(\text{Customer} * (\text{Sale} * (\sigma_{\text{Author = 'Pratchett'}}\text{Book})))$
   b.
```
SELECT Fname, Lname, Title
FROM  Sale s
  JOIN Book_Sold bs on s.Sale_ID = bs.Sale_ID
  JOIN Customer c on c.Customer_ID = s.Customer_ID
  JOIN Book b on b.ISBN = bs.ISBN
  JOIN Book_Author ba on b.ISBN = ba.ISBN
WHERE ba.Name LIKE "%Pratchett%";
```

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

5. This query finds the customer who has purchased the most books and the total number of books that they have purchased.
   a. $\pi_{\text{Name, number\_sales}}(\sigma_{\text{number\_sales = max\_sold}}(\text{Customer} * ((_{\text{Customer\_ID}}\mathcal{F}_{\text{COUNT Sale\_ID AS number\_sales}}(\text{Sale})) * \mathcal{F}_{\text{MAX Count\_Sale\_ID AS max\_sold}}(_{\text{Customer\_ID}}\mathcal{F}_{\text{COUNT Sale\_ID AS Count\_Sale\_ID}}(\text{Sale}))))$
   b.
```
SELECT Fname, Lname, SUM(Quantity) AS Total_Books_Purchased
From Sale s
        JOIN Book_Sold bs on s.Sale_ID = bs.Sale_ID
        JOIN Customer c on c.Customer_ID = s.Customer_ID
GROUP BY s.Customer_ID
ORDER BY Total_Books_Purchased DESC
LIMIT 1;
```

6. This query counts the number of books in the system by each publisher.
   a. $_{\text{Publisher\_ID}}\mathcal{F}_{\text{Name, COUNT ISBN}}(\text{Book} * \text{Publisher})$
   b.
```
SELECT Publisher, COUNT(Book.ISBN) AS Book_Count
FROM Book GROUP BY Publisher
```

7. This query counts how many promotions an employee has created.
   a. $_{\text{Employee\_Number}}\mathcal{F}_{\text{Name, COUNT Promotion\_ID}}(\text{Employee} * \text{Promotion})$
   b.
```
SELECT Fname, Lname, COUNT(p.Promotion_ID) AS Number_Of_Promotions
FROM Employee e
  JOIN Promotions p on e.Employee_Number = p.Employee_Number
GROUP BY e.Employee_Number
```

8. This query finds the total number of times each book has been sold.
   a. $_{\text{ISBN}}\mathcal{F}_{\text{Title, COUNT Quantity}}(\text{Sale} * \text{Book})$
   b.
```
SELECT Title, SUM(Quantity) AS Total_Sold
FROM Book_Sold bs
  JOIN Book b on b.ISBN = bs.ISBN
GROUP BY b.ISBN
```

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Advanced Queries (Relational Algebra not Required)

1. This query provides a list of customer names, along with the total dollar amount each customer has spent.

```
SELECT Fname, Lname, ROUND(SUM(bs.Quantity * b.Price), 2) AS Total_Spent
FROM Sale S
JOIN Customer C on S.Customer_ID = C.Customer_ID
JOIN Book_Sold BS on S.Sale_ID = BS.Sale_ID
JOIN Book B on B.ISBN = BS.ISBN
GROUP BY c.Customer_ID
```

2. This query provides a list of customer names and e-mail addresses for customers who have spent more than the average customer.

```
SELECT C.Fname, C.Lname, C.Email
FROM Customer C
JOIN Sale S on C.Customer_ID = S.Customer_ID
JOIN Book_Sold BS on S.Sale_ID = BS.Sale_ID
JOIN Book B on B.ISBN = BS.ISBN
GROUP BY C.Customer_ID
HAVING SUM(BS.Quantity * B.Price) > (
  SELECT AVG(Total_Per_Customer)
  FROM (
    SELECT C2.Customer_ID, SUM(BS2.Quantity * B2.Price) AS Total_Per_Customer
    FROM Customer C2
    JOIN Sale S2 on C2.Customer_ID = S2.Customer_ID
    JOIN Book_Sold BS2 on S2.Sale_ID = BS2.Sale_ID
    JOIN Book B2 on B2.ISBN = BS2.ISBN
    GROUP BY C2.Customer_ID
     ))
```

3. This query provides 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 least.

```
SELECT B.Title, SUM(BS.Quantity) AS Total_Sold
FROM Book_Sold BS
JOIN Book B on B.ISBN = BS.ISBN
GROUP BY BS.ISBN
ORDER BY Total_Sold DESC
```

4. This query provides 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 B.Title, ROUND(SUM(B.Price * BS.Quantity), 2) AS Total_Sales
   FROM Book_Sold BS
   JOIN Sale S on S.Sale_ID = BS.Sale_ID
   JOIN Book B on B.ISBN = BS.ISBN
   GROUP BY BS.ISBN
   ORDER BY Total_Sales DESC
   ```

5. This query finds the most popular author in the database (i.e. the one who has sold the most books)

   ```
   SELECT BA.Name, SUM(BS.Quantity) AS Total_Books_Sold
   FROM Book_Sold BS
   JOIN Book_Author BA on BS.ISBN = BA.ISBN
   GROUP BY BA.Name
   ORDER BY Total_Books_Sold DESC
   LIMIT 1
   ```

6. This query finds the most profitable author in the database for this store (i.e. the one who has brought in the most money)

   ```
   SELECT BA.Name, ROUND(SUM(BS.Quantity * B.Price), 2) AS Total_Sales
   FROM Book_Sold BS
   JOIN Book_Author BA on BS.ISBN = BA.ISBN
   JOIN Book B on B.ISBN = BS.ISBN
   GROUP BY BA.Name
   ORDER BY Total_Sales DESC
   LIMIT 1
   ```

7. This query provides a list of customer information for customers who purchased anything written by the most profitable author in the database.

   ```
   SELECT DISTINCT C.Customer_ID, C.Fname, C.Lname, C.Street_Address, C.City, C.State, C.Zip,
   C.Email, C.Phone_Number
   FROM Customer C
   JOIN Sale S on C.Customer_ID = S.Customer_ID
   JOIN Book_Sold BS on S.Sale_ID = BS.Sale_ID
   JOIN Book B on BS.ISBN = B.ISBN
   JOIN Book_Author BA on BS.ISBN = BA.ISBN
   ```

```
JOIN (
   SELECT BA.Name, ROUND(SUM(BS.Quantity * B2.Price), 2) AS Total_Sales
   FROM Book_Sold BS
   JOIN Sale S on BS.Sale_ID = S.Sale_ID
   JOIN Book_Author BA on BS.ISBN = BA.ISBN
   JOIN Book B2 on B2.ISBN = BS.ISBN
   GROUP BY BA.Name
   ORDER BY Total_Sales DESC
   LIMIT 1
) AS Most_Profitable_Author ON Most_Profitable_Author.Name = BA.Name
```

8. This query provides the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
SELECT DISTINCT BA.Name
FROM Book_Author BA
JOIN Book_Sold BS3 on BA.ISBN = BS3.ISBN
JOIN Sale S3 on S3.Sale_ID = BS3.Sale_ID
JOIN (
   SELECT C.Customer_ID
   FROM Customer C
   JOIN Sale S on C.Customer_ID = S.Customer_ID
   JOIN Book_Sold BS on S.Sale_ID = BS.Sale_ID
   JOIN Book B on B.ISBN = BS.ISBN
   GROUP BY C.Customer_ID
   HAVING SUM(BS.Quantity * B.Price) > (
   SELECT AVG(Total_Per_Customer)
   FROM (
      SELECT C2.Customer_ID, SUM(BS2.Quantity * B2.Price) AS Total_Per_Customer
      FROM Customer C2
      JOIN Sale S2 on C2.Customer_ID = S2.Customer_ID
      JOIN Book_Sold BS2 on S2.Sale_ID = BS2.Sale_ID
      JOIN Book B2 on B2.ISBN = BS2.ISBN
      GROUP BY C2.Customer_ID
       )
   )
) AS Customers_Spent_Above_Average ON Customers_Spent_Above_Average.Customer_ID =
S3.Customer_ID
```

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Inserts

1. INSERT INTO Book (ISBN, Title, Price, Genre, Publication_Year, Publisher)
   VALUES ('9780553573404', ‘Harry Potter’, 9.99, 'Fantasy', 1996, ‘Publisher name');

2. INSERT INTO Book_Author (ISBN, Name)
   VALUES ('9780553573404', 'George R.R. Martin');

3. INSERT INTO Book_Sold (Sale_ID, ISBN, Quantity)
   VALUES (1001, '9780553573404', 3);

4. INSERT INTO Customer (Customer_ID, Fname, Lname, Street_Address, City, State, Zip, Email, Phone_Number, Password)
   VALUES (1001, 'John', 'Doe', '123 Main St', 'Anytown', 'CA', 12345, 'johndoe@example.com', 5555551234, 'password123');

5. INSERT INTO Employee (Employee_Number, Fname, Lname, Street_Address, City, State, Zip, Email, Phone_Number, Salary, Title, Hire_Date, Password)
   VALUES (1001, 'Jane', 'Smith', '456 Oak St', 'Anytown', 'CA', 12345, 'janesmith@example.com', 5555555678, 50000, 'Sales Associate', 20220101, 'password456');

6. INSERT INTO Promotions (Promotion_ID, ISBN, Discount, Begin_Date, End_Date, Employee_Number)
   VALUES (1001, '9780553573404', 0.2, 20220501, 20220601, 1001);

7. INSERT INTO Sale (Sale_ID, Date, Credit_Card_Number, Customer_ID)
   VALUES (1001, 20220501, '1234-5678-9012-3456', 1001);

   If there are dependencies in the system, such as a foreign key constraint that requires a record to exist in another table before it can be inserted, then the records should be inserted in the appropriate order to avoid violating constraints. For example, a new Sale record cannot be inserted before a corresponding Customer record exists, since the Customer_ID field in Sale is a foreign key to the Customer table. Also, a promotion cannot be added without the employee that created it being in the system. A book sold cannot be added until the sale tuple has been created that it will be a part of and the book that is being sold is in the system.

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Deletes

1. **Book**
   DELETE FROM Book
   WHERE ISBN = ' 782140661';

2. **Book_Author**
   DELETE FROM Book_Author
   WHERE Name = ‘Chip Dawes’;

3. **Book_Sold**
   DELETE FROM Book_Sold
   WHERE Sale_ID = 19;

4. **Customer**
   DELETE FROM Customer
   WHERE Customer_ID = 12;

5. **Employee**
   DELETE FROM Employee
   WHERE Employee_Number = 10;

6. **Promotions**
   DELETE FROM Promotions
   WHERE Promotion_ID = 9;

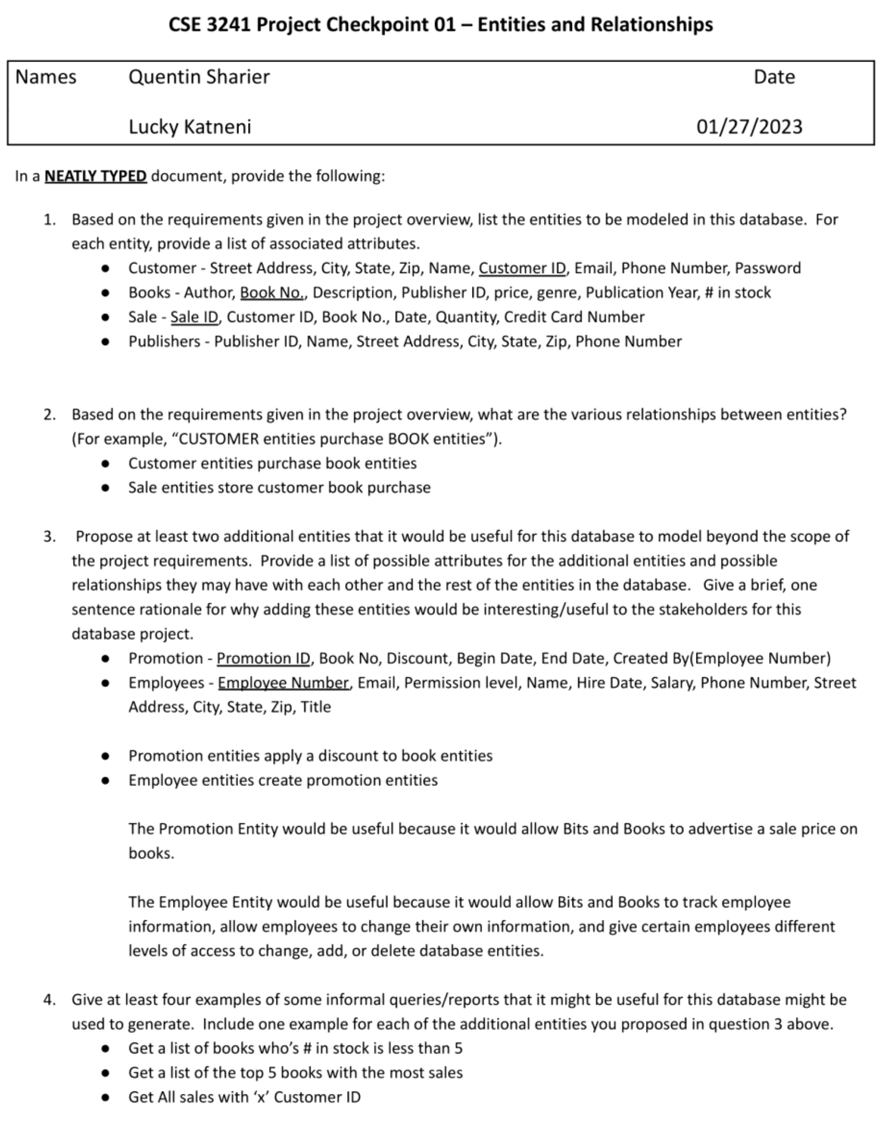
7. **Sale**
   DELETE FROM Sale
   WHERE Sale_ID = 14;

   If there are dependencies in the system, such as a foreign key constraint, it is important to delete rows in the correct order to avoid violating referential integrity. For example, if we delete an employee, we must delete all promotions that the employee created first. If we need to delete a customer from the database, we should first delete any sales associated with that customer from the Sale table, and then delete the customer. An example of this can be seen below:

Appendix

Graded Checkpoints

1. Checkpoint 1



CSE 3241 Project Checkpoint 01 – Entities and Relationships

Names	Quentin Sharier	Date
	Lucky Katneni	01/27/2023

In a **NEATLY TYPED** document, provide the following:

1. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes.
   - Customer - Street Address, City, State, Zip, Name, Customer ID, Email, Phone Number, Password
   - Books - Author, Book No., Description, Publisher ID, price, genre, Publication Year, # in stock
   - Sale - Sale ID, Customer ID, Book No., Date, Quantity, Credit Card Number
   - Publishers - Publisher ID, Name, Street Address, City, State, Zip, Phone Number

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, "CUSTOMER entities purchase BOOK entities").
   - Customer entities purchase book entities
   - Sale entities store customer book purchase

3. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief, one sentence rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.
   - Promotion - Promotion ID, Book No, Discount, Begin Date, End Date, Created By(Employee Number)
   - Employees - Employee Number, Email, Permission level, Name, Hire Date, Salary, Phone Number, Street Address, City, State, Zip, Title

   - Promotion entities apply a discount to book entities
   - Employee entities create promotion entities

   The Promotion Entity would be useful because it would allow Bits and Books to advertise a sale price on books.

   The Employee Entity would be useful because it would allow Bits and Books to track employee information, allow employees to change their own information, and give certain employees different levels of access to change, add, or delete database entities.

4. Give at least four examples of some informal queries/reports that it might be useful for this database might be used to generate. Include one example for each of the additional entities you proposed in question 3 above.
   - Get a list of books who's # in stock is less than 5
   - Get a list of the top 5 books with the most sales
   - Get All sales with 'x' Customer ID

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

- Get all books with the genre “Chemistry”
- Get the Discount from promotion table with the Book foreign ID of ‘x’
- Get the Permission level of the employee with ID ‘x’

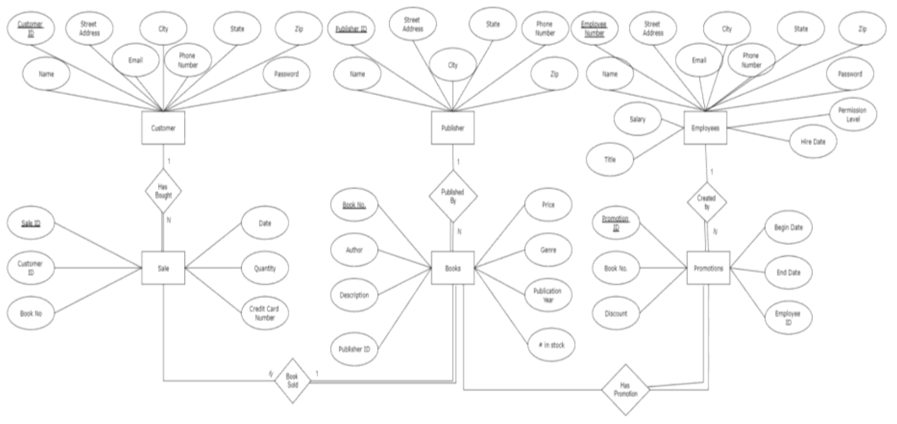
5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you’ve outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities.

    To add a new publisher, you would only need to add it to the publisher table with a unique publisher ID. With this method, it is possible to add a new publisher without knowing the books that they publish. The publisher will be added to the books they published when the book gets added to the database.

6. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 3 above.

    1. Update # in stock in the Books table with Book No. ‘x’ to (# in stock - 1) No other attributes need altered.
    2. Update permission level in the employee table with employee number ‘x’ to ‘2’ No other attributes need altered.
    3. Update the End Date in the promotion table with promotion number ‘x’ to ‘02/04/2023’ No other attributes need altered

7. Provide an ER diagram for your database. Make sure you include all of the entities and relationships you determined in the questions above ***INCLUDING the entities for question 3 above***, and remember that ***EVERY*** entity in your model needs to connect to another entity in the model via some kind of relationship.



Feedback: None given.

Revision: No revision needed.

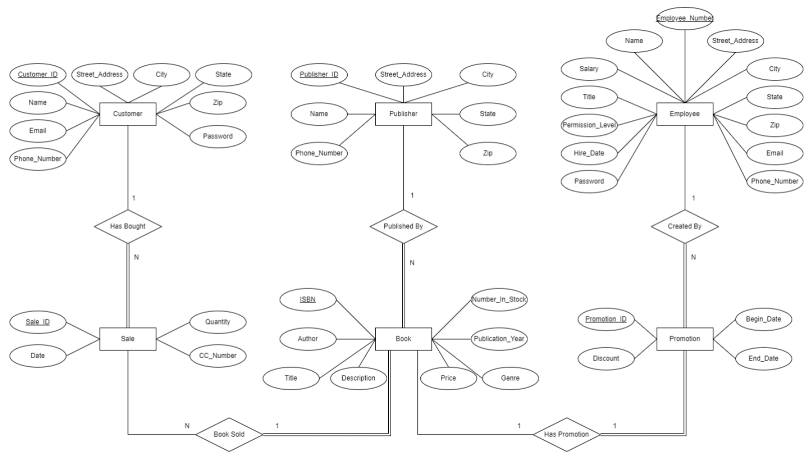
2. Checkpoint 2

CSE 3241 Project Checkpoint 02 – Relational Model and Relational Algebra

Names	Quentin Sharier	Date
	Lucky Katneni	02/21/23

In a **NEATLY TYPED** document, provide the following:

1. Provide a current version of your ER Model as per Project Checkpoint 01. If you were instructed to change the model for Project Checkpoint 01, make sure you use the revised version of your ER Model.



2. Map your ER model to a relational schema. Indicate all primary and foreign keys.

“ “ = Foreign Key

- Customer(Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password)
- Sale(Sale_ID, Date, Quantity, Credit_Card_Number, Customer_ID, ISBN)
- Book(ISBN, Author, Title, Description, Price, Genre, Publication_Year, Number_In_Stock, Publisher_ID)
- Publisher(Publisher_ID, Name, Street_Address, City, State, Phone_Number, Zip)
- Employees(Employee_Number, Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password)
- Promotions(Promotion_ID, Book_No, Discount, Begin_Date, End_Date, Employee_Number, ISBN)

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

3. Given your relational schema, provide the relational algebra to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries:
    a. Find the titles of all books by Pratchett that cost less than $10

    $\pi_{\text{Title}}(\sigma_{\text{Author='Pratchett'} \wedge \text{Price} < 10}(\text{Book}))$

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

    $\pi_{\text{Title, Date}}(\text{Book} * (\sigma_{\text{Customer\_ID} = 453}\ \text{Sale}))$

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

    $\pi_{\text{Title, ISBN}}(\sigma_{\text{Number\_In\_Stock} < 5}(\text{Book}))$

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

    $\pi_{\text{Name, Title}}(\text{Customer} * (\text{Sale} * (\sigma_{\text{Author = 'Pratchett'}}\ \text{Book})))$

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

    $\mathcal{F}_{\text{COUNT Sale\_ID}}(\sigma_{\text{Customer\_ID} = 251}\ \text{Sale})$

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

    $\pi_{\text{Name, Count\_Sale\_ID}}(\sigma_{\text{Count\_Sale\_ID} = \text{MAX Count\_Sale\_ID}}(\text{Customer} * ({}_{\text{Customer\_ID}}\mathcal{F}_{\text{COUNT Sale\_ID}}(\text{Sale}))))$

4. Come up with three additional interesting queries that your database can provide. Give what the queries are supposed to retrieve in plain English and then as relational algebra. Your queries should include joins and at least one should include an aggregate function. At least one of your queries should use "extra" entities you added to your model in Checkpoint 01.

    1. A query that counts the number of books in the system by each publisher.

    ${}_{\text{Publisher\_ID}}\mathcal{F}_{\text{Name, COUNT ISBN}}(\text{Book} * \text{Publisher})$

    2. A query that counts how many promotions an employee has created.

    ${}_{\text{Employee\_Number}}\mathcal{F}_{\text{Name, COUNT Promotion\_ID}}(\text{Employee} * \text{Promotion})$

    3. A query that finds the total number of times each book has been sold.

    ${}_{\text{ISBN}}\mathcal{F}_{\text{Title, COUNT Quantity}}(\text{Sale} * \text{Book})$

Feedback: “3f. Aggregation operation MAX cannot be used directly in the condition of select operation.”

Revision: A revision to this, to question 3f, can be seen on Page 12 query #5.

3. Checkpoint 3

   Please see the “ProjectCheckpoint3” folder inside the “Documentation” folder for project checkpoint 3.

   Feedback: None given.

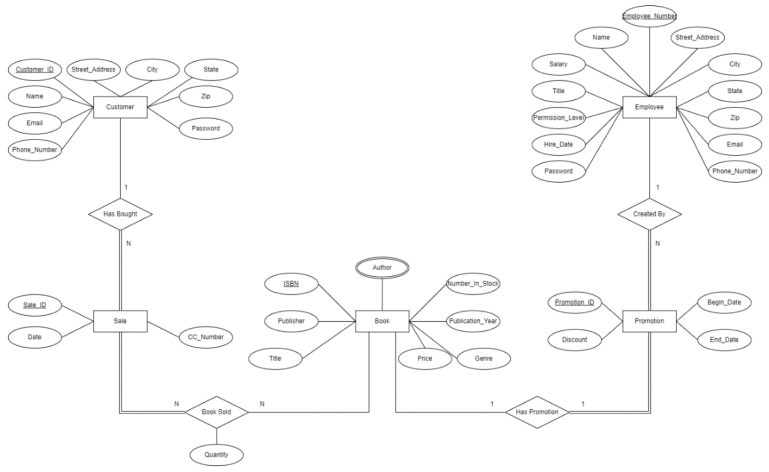
   Revision: No revision needed.

4. Checkpoint

Project Checkpoint 4

Quentin Sharier
Lucky Katneni
04/18/2023
CSE 3241

1.



“ “ = Foreign Key

- Customer(<u>Customer_ID</u>, Fname, Lname, Street_Adress, City, State, Zip, Email, Phone_Number, Password)
- Sale(<u>Sale_ID</u>, Date, Credit_Card_Number, Customer_ID)
- Book_Sold(<u>Sale_ID</u>, <u>ISBN</u>, Quantity)
- Book(<u>ISBN</u>, Title, Price, Genre, Publication_Year, Number_In_Stock, Publisher)
- Book_Author(<u>ISBN</u>, <u>Name</u>)
- Employees(<u>Employee_Number</u>, Fname, Lname, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password)
- Promotions(<u>Promotion_ID</u>, Discount, Begin_Date, End_Date, Employee_Number, ISBN)

2.

Customer

- {Customer_ID} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }
- {Email} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }
- {Phone_Number} -> { Customer_ID, Name, Street_Adress, City, State, Zip, Email, Pone_Number, Password }

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Project Checkpoint 4

Quentin Sharier
Lucky Katneni
04/18/2023
CSE 3241

Sale
- {Sale_ID} -> { Sale_ID, Date, Credit_Card_Number, Customer_ID }

Book Sold
- {Sale_ID, ISBN} -> {Quantity}

Book
- {ISBN} -> { Title, Price, Genre, Publication_Year, Number_In_Stock, Publisher }

Book Author
- {ISBN} -> {Name}

Employees
- {Employee_Number} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }
- {Email} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }
- {Phone_Number} -> { Name, Street_Adress, City, State, Zip, Email, Phone_Number, Salary, Title, Permission_Level, Hire_Date, Password }

Promotions
- {Promotion_ID} -> { Discount, Begin_Date, End_Date, Employee_Number, ISBN }

3.

Customer – BCNF
Sale - BCNF
Book_Sold - BCNF
Book - BCNF
Book_Author - BCNF
Employees - BCNF
Promotions - BCNF

All relations are in BCNF thus, in 3rd normal form as well.

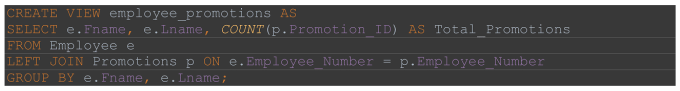
4.

All relations are in BCNF.

5.

View 1

This view shows the total number of promotions each employee has been involved in, including those who have not been involved in any promotions.



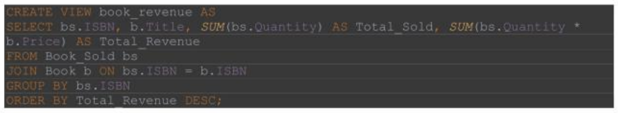
```sql
CREATE VIEW employee_promotions AS
SELECT e.Fname, e.Lname, COUNT(p.Promotion_ID) AS Total_Promotions
FROM Employee e
LEFT JOIN Promotions p ON e.Employee_Number = p.Employee_Number
GROUP BY e.Fname, e.Lname;
```

Project Checkpoint 4

Quentin Sharier
Lucky Katneni
04/18/2023
CSE 3241

View 2

This view calculates the total number sold and total revenue generated by each book. It also orders the books from most revenue generated to least.



```sql
CREATE VIEW book_revenue AS
SELECT bs.ISBN, b.Title, SUM(bs.Quantity) AS Total_Sold, SUM(bs.Quantity *
b.Price) AS Total_Revenue
FROM Book_Sold bs
JOIN Book b ON bs.ISBN = b.ISBN
GROUP BY bs.ISBN
ORDER BY Total_Revenue DESC;
```

Quentin Sharier
Lucky Katneni
CSE 3241
04/27/2023

Feedback: “2. I think you may miss the following dependencies: Sale: Many attributes may depend on credit_card_number Employee: Title -> Permission_Level 3. Customer, Sale, and Employee is in 2NF instead of 3NF. Please modify your relational model accordingly.”

Revision: No revision needed after discussion with professor. A credit card number can be used by multiple customers. The employee’s title does not dictate what level they are trained on in the database. This makes Sale in 3NF. Customer and Employee are in 3NF because they are in 2NF, and no nonprime attributes are transitively dependent on the primary keys. Customer_ID and Employee_ID are the primary keys while Email and Phone_Number are both candidate keys, thus the transitivity rule does not apply to them as they are not non-prime.