# CSE 3241 FINAL PROJECT REPORT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted To:**

Jeffrey Eden

**Created By:**

Group 08

Han Han

Qiheng Chen

Tongshuai Zhang

Yuliang Fu

Computer Science Engineering 3241

The Ohio State University

Columbus, OH

05 November 2018

## **SECTION 1: DATABASE DESCRIPTION:**

### A: Database description document that contains the following information about your database(compiled from your completed and revised checkpoints): professionally presented, well-formatted ER-model that reflects the updates you have made during the semester.

### **B. The professionally presented, well-formatted relational schema for the database.**

**(Note: Red color means the foreign key and the underline means the primary key)**

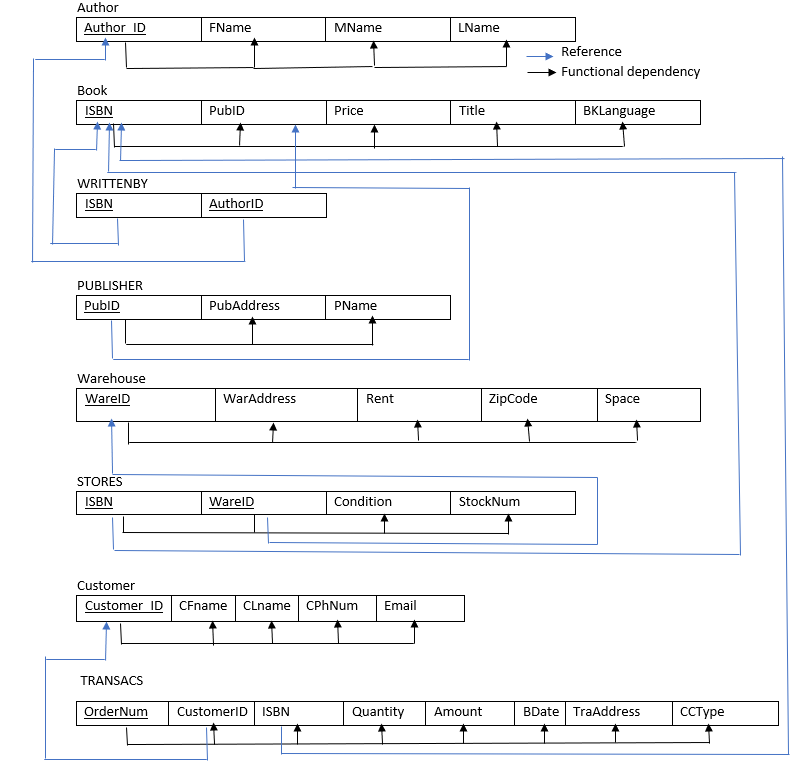
**Relational schema Conceptual Model:**

* **AUTHOR(**AuthorID , Fname, Lname, Mname)
* **BOOK**: (ISBN, **PubID,** Price, Title, BKLanguage)
* **WRITTENBY: (ISBN, AuthorID)**
* **PUBLISHER**: (PubID, PubAddress, PName)
* **WAREHOUSE: (**WareID**,** WarAddress, Rent, ZipCode, Space)
* **STORES: (ISBN ,WareID,** Condition, StockNum)
* **CUSTOMER: (**CustomerID,CFname,CLname,CPhNum, Email)
* **TRANSACS: (**OrderNum, **CustomerID**, **ISBN,** Quantity, Amount, BDate, TraAddress, CCType)

**Functional Dependencies:**

* FD1: ***AUTHOR***: **{** **AuthorID } → {** Fname, Lname, Mname **}**
* FD2: ***BOOK*: { ISBN } →** **{** Price, Title, BKLanguage, PubID, PubAddress, PName **}**
* FD3: ***WRITTENBY*: { ISBN, AuthorID } → {}**
* FD4: ***PUBLISHER*: { PubID } →** **{** PubAddress, PName **}**
* FD5: ***WAREHOUSE:* { WareID }→ {** WarAddress, Rent, Space, ZipCode **}**
* FD6: ***STORES:* { ISBN, WareID } → {** Condition, StockNum **}**
* FD7:***CUSTOMER:* { CustomerID } → {** Email,CFname,CLname,CPhNum **}**
* FD8: ***TRANSACS:* { OrderNum** **}→{** CustomerID, ISBN, Quantity, Amount, BDate, TraAddress, CCType**}**

**Neat-Complete Relational Schema:**

****

### **C. Normalization Description of each level of achieved for all tables**

**Demonstration for justifying this is in BCNF!**

**it’s necessary to build a 2NF for further demonstration and usage.**

**For the AUTHOR relation schema, the AuthorID attribute can determine the other three non-prime attributes that are all *fully functionally dependent* the primary key *AuthorID.***

For the **BOOK relation schema,** the ISBN attribute can determine the other three non-prime attributes that are all *fully functionally dependent* the primary key *ISBN.*

For the **WRITTENBY schema**, the combination of ISBN, AuthorID forming the primary key can determine the other non-prime attributes that are all *fully functionally dependent on* them*.*

For the **PUBLISHER relation schema**, the PubID or PubAddress or both two attributes can determine the other non-prime attribute that is all *fully functionally dependent on* the primary key *PubID.* So, it’s in 3NF.

Same analysis for the **WAREHOUSE schema**.

For the **STORES relation schema,** the primary key: ISBN, WareID can determine the other two non-prime attributes that are all *fully functionally dependent* on them*.*

For the **TRANSACTION relation schema,** the primary key OrderNum can determine the other two non-prime attributes that are all *fully functionally dependent on* them*.* No other non-primary attributes can determine the other non-primary attributes.

Besides, no non-prime attributes of the ten relations are transitively dependent on the primary key. All the relations are in 3NF.

All the relation schemas **are in BCNF,** the only analysis is to demonstrate the STORES relation schema since it has the superKey. However, there is no dependency that the non-prime attributes can determine the prime attribute. So, the schema should be in 3NF.

### **D. Indexes Description that you have chosen to implement on your database, along with rationale for each.**

#### **Use B-Trees index approach for the Column ‘Price’ of Table ‘Book’:**

It is useful for searching in a range of book prices as most of the online bookstore has the functionality called filter which can sort the book by the price in a given range defined by the customer. Besides, the query is normally based on specific product with additional requirements so it’s necessary to use the approach which can handle multiple columns hence include multiple references. Therefore, B-Trees is the most appropriate index way to implement the index of Column ‘Price’ of Table ‘Book’. Furthermore, the delete and insert cost lower time.

#### **Use Clustering index approach for the Column ‘StockNum’ of Table ‘STORES’**

Usually, for customers, the online bookstore would remind customers about the stock condition, which is defined into three parts, out of stock, low-stock, and in-stock. This function can be realized by clustering index. For example, we can use stock number 0, 0-10,and >10 as the condition of indexing. After implementing the clustering index, the new table is pretty clear for the owner of the bookstore, reminding him/her to restock books. The reason for choosing the clustering index is that it’s easy to implement the table and we do not need to insert and delete the STORES table frequently.

#### **Use Hash-based index approach for the Column ‘BKCondition’ of Table ‘STORES’**

Most bookstores can inform customers about the conditions such as new, and old. When we arrange the STORES table by using the hash-based index, customers can easily see what condition of books are in a short request time due to the hash function design. The database does not need a lot of insert/delete actions, because the condition is usually fixed when the data was imported initially. Therefore, the Hash-based indexing is appropriate.

#### **Use B-Trees index approach for the Column ‘Quantity’ of Table ‘TRANSACTS’:**

After implementing the B-Trees, the new table is generated by the attribute of Quantity in TRANSACTS. This table can help the bookstore to understand which book was sold well and accordingly, they can change the strategy of selling books to have more benefits in the future. The Quantity values change a lot due to the attribute of the transaction. The database needs a lot of insert/delete actions. Therefore, the B-Trees indexing is appropriate.

#### **Use B-Trees index approach for the Column ‘Amount’ of Table ‘TRANSACTS’:**

After implementing the B-Trees of Amount in TRANSACTS, people can have a new table which can show customer information by the amount they spent. People can distinguish valued customers who spent most in the past. According to how much customers they spent, the bookstore can give some certain discounts or making targeted advertising. The database needs a lot of insert/delete actions. Therefore, the B-Trees indexing is appropriate.

### **E.** **View Description**

#### **VIEW01: Highest sold amount(most money) Book by each Author**

This view displays a list of most popular book(sold most money) ISBN of each author with the book ISBN, Title, sold amount, sold quantity, price

**Relation Schema**

R1A: (BOOK \* AUTHOR\* WRITTENBY\*TRANSACTS)

R1B:π(AuthorID,FName,LName,ISBN,Title, (ISBN Fas AmountSold), (ISBN Fas QuantitySold)R1A

Result:π(AuthorID,FName,LName,ISBN,Title, (AuthorID Fas AmountSold),QuantitySold, SinglePrice)R1B

**SQL Statements**

CREATE VIEW Best\_BOOK\_EachAuthor

AS SELECT AuthorID, ISBN, Title, MAX(AmountSold) AS AmountSold, QuantitySold, SinglePrice

FROM (SELECT A.AuthorID,A.FName,A.LName,B.ISBN,B.Title,SUM(T.Amount) AS AmountSold, SUM(T.Quantity) AS QuantitySold,B.Price AS SinglePrice

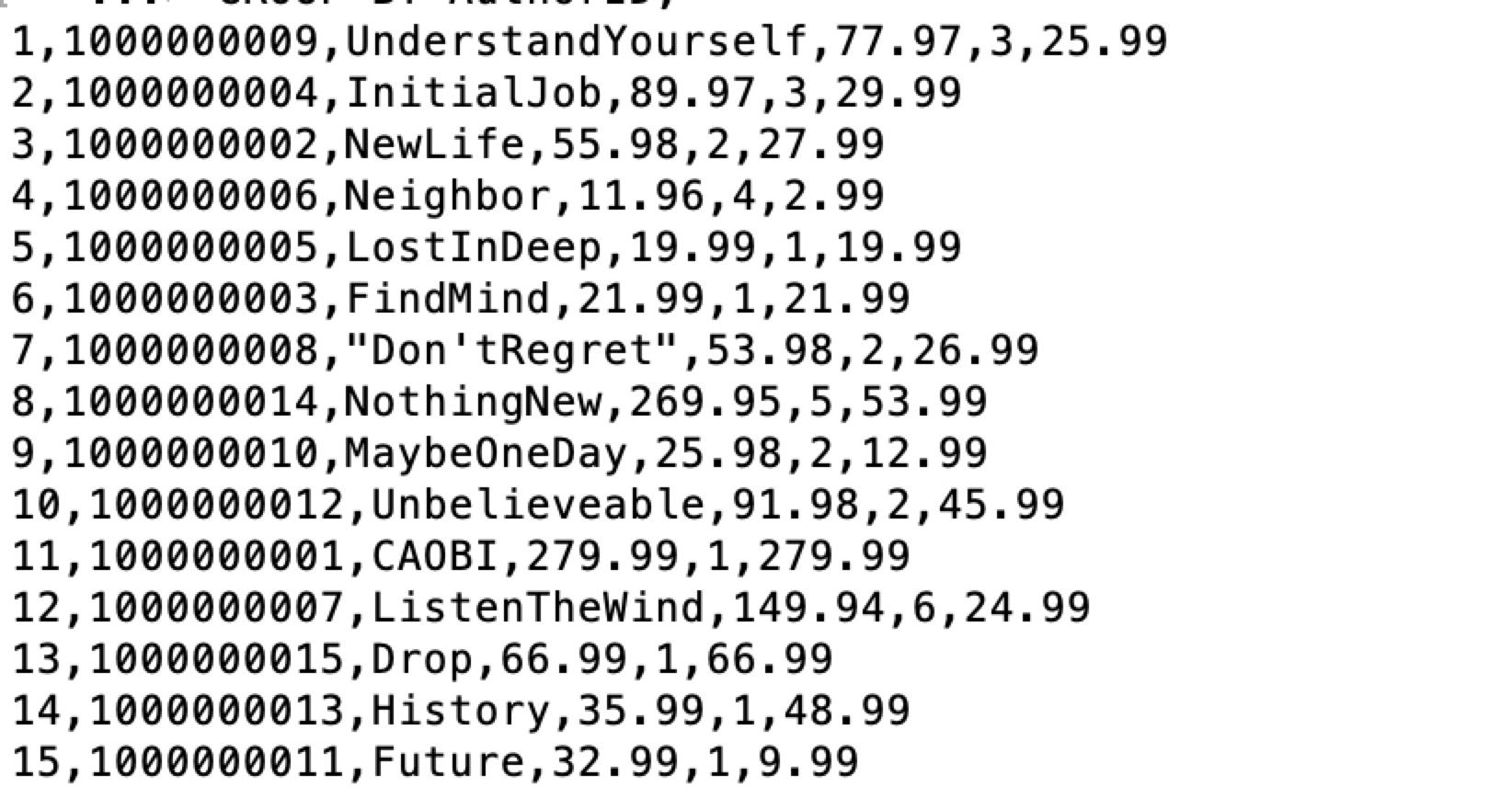
from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T

WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID AND B.ISBN= T.ISBN

GROUP BY B.ISBN) AS Author\_sold

GROUP BY AuthorID;

**Sample Output**



#### **VIEW02:Total Spent by each Customer**

This view displays the total amount paid by each customer

including all the transactions in the system.

**Relation Schema**

R1A: (CUSTOMER \* TRANSACTS)

Result :π(C.CustomerID,C.CFName,C.CLName, (C.CustomerID Fas TotalConsume))R1A

**SQL Statements**

CREATE VIEW Total\_Consume\_EachCustomer

AS SELECT C.CustomerID,C.CFName,C.CLName,SUM(T.Amount) AS TotalConsume

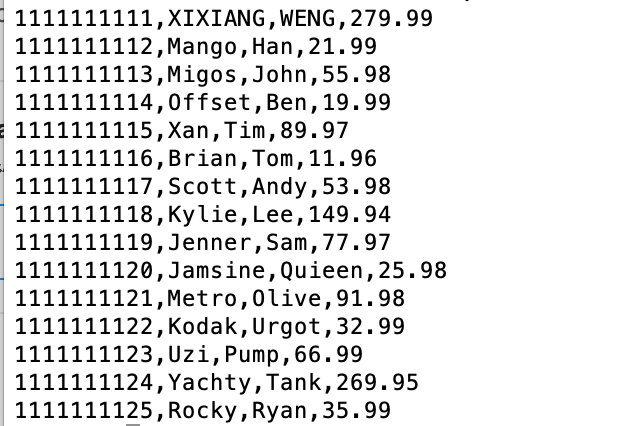
FROM CUSTOMER C, TRANSACTS T

WHERE C.CustomerID = T.CustomerID

GROUP BY C.CustomerID

ORDER BY C.CustomerID;

**Sample Output**

****

### 

### **F. Sample transactions**

#### **1. Add an author to the database**

a. This is useful because it adds a new author to the database without not knowing if the author has written any books stored in the bookstore database.

**BEGIN TRANSACTION ADD\_NEW\_AUTHOR**

**INSERT INTO AUTHOR VALUES (123, 'TONGSHUAI','kkk','ZHANG');**

**IF error THEN GO TO UNDO; END IF;**

**COMMIT;**

**GO TO FINISH;**

**UNDO:**

**ROLLBACK;**

**FINISH:**

**END TRANSACTION**

#### **2. Add 50 new copies of book "0000000002" to Warehouse 001**

This would be useful when you want to add more books to a certain warehouse, this also makes it easier to keep track of which warehouse each book is in.

**BEGIN TRANSACTION ADDBOOKS\_WAREHOUSE**

**UPDATE STORES**

**SET StockNum= StockNum + '50'**

**WHERE WareID = 001**

**IF error THEN GO TO UNDO;**

**END IF;**

**COMMIT;**

**GO TO FINISH;**

**UNDO:**

**ROLLBACK;**

**FINISH:**

**END TRANSACTION**

#### **3. A new customer signs up**

This is useful because new users sign up who have not ordered any books

**BEGIN TRANSACTION ADD\_NEW\_CUSTOMER**

**INSERT INTO CUSTOMER VALUES (1111111111,"XIXIANG","WENG","6146804976","zhang.7686@osu.com"),**

**IF error THEN GO TO UNDO; END IF;**

**COMMIT;**

**GO TO FINISH;**

**UNDO:**

**ROLLBACK;**

**FINISH:**

**END TRANSACTION**

## **SECTION 2: USER MANUAL**

### **A. Description of each entity and its attributes, including its data type and any constraints you have built-in.**

#### AUTHOR – description of the author of the book

**This table is used to show the author of the book. LName is the author’s last name; MName is the author’s middle name; FName is the author’s first name; AuthorID is the primary key in this relation, it is used to identify each author, users can check each author by AuthorID.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| AuthorID | INTEGER | Unique identifier for an author(Primary Key) |
| Lname | VARCHAR | Author’s last name |
| Mname | VARCHAR | Author’s middle name |
| Fname | VARCHAR | Author’s first name |

**Note: Middle name is set to “No middle” as default.**

#### BOOK – description of the book

**This table gives some basic attribute of a book. ISBN is the Identifier of the book. This is the primary key, users can find the book by using ISBN; PubID is the foreign key to the publisher, user can find the book’s PUblisher by using PubID; Price is the price of this book; Title is the name of the book; BKlanguage is about what language the book is in.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| ISBN | CHAR | Unique Identifier of a book(Primary Key) |
| PubID | INTEGER | Foreign KEY to PUBLISHER |
| Price | VARCHAR | Price of the book |
| Title | VARCHAR | Title of the book |
| BKLanguage | VARCHAR | The language of the book |

#### WRITTENBY- relationship between a book and the author

**This view helps connect the book and the author. Both of the Attributes are the foreign key to BOOK and Author. By using it, user has the relationship to connect BOOK and AUTHOR.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| ISBN | CHAR | Foreign key to BOOK |
| AuthorID | INTEGER | Foreign key to AUTHOR |

**Note: ISBN and AuthorID will be combined as the primary key**

#### PUBLISHER – description of the publisher of the book

**This view helps to track the publisher of a book. PubID is the primary key which is able to help the user to find the book by using it; PubAddress is the address of the publisher; Pname is the name of the publisher.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| PubID | INTEGER | Unique identifier of the publisher |
| PubAddress | VARCHAR | Address of the publisher |
| PName | VARCHAR | Publisher’s name |

#### WAREHOUSE- description of the warehouse of the book

**This table helps to know the warehouse situation. WareID is the primary key of the warehouse. Users can find the warehouse by using the WareID. WareAddress is the address of a warehouse; Rent is the renting price for this warehouse; ZipCode is the ZIP Code of this warehouse; Space is how big this space is.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| WareID | INTEGER | Identifier for the warehouse |
| WareAddress | VARCHAR | Address of the warehouse |
| Rent | DOUBLE | Renting price of the warehouse |
| ZipCode | CHAR | ZIP code of the warehouse |
| Space | DOUBLE | The space the warehouse has |

#### STORES:- relationship between a book and the warehouse

**This table helps to connect the book and the warehouse. ISBN and WareID are primary key of the table; ISBN is the foreign key to BOOK and WareID is the foreign key to WAREHOUSE. Condition is the book’s condition in this warehouse; StockNum is the number in stock of this book.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| ISBN | CHAR | Foreign key to BOOK |
| WareID | INTEGER | Foreign key to WAREHOUSE |
| Condition | VARCHAR | The book condition(good or bad, new or used) |
| StockNum | INTEGER | The number in stock of the book |

**Note: ISBN and WareID will be combined as the primary key**

#### CUSTOMER – description of the customer of the book

**This table helps to track the customer.CLName is the customer’s last name, CMName is the customer’s middle name,CFName is the customer’s first name; CPhNum is the customer’s phone number; Email is the customer’s email address; CustomerID is the primary key in this relation, it is used to identify each customer, user can check each customer by customerID .**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| CustomerID | INTEGER | Unique identifier of a customer |
| CFname | VARCHAR | Customer’s first Name |
| CLname | VARCHAR | Customer’s last Name |
| CPhNum | CHAR | Customer’s phone number |
| Email | VARCHAR | Customer’s email |

**Note: Middle name is set to Nomiddle as default.**

#### TRANSACTS - relationship between a book and the purchaser(customer)

**This view helps connect the book and the customer. OrderNum is the primary key of each transacts, user can find each transact by using OrderNum; CustomerID is the foreign key to CUSTOMER and ISBN is the foreign key to BOOK; Quantity is how much books a customer buy in a single order; Amount is the total price of this order; BDate is the order date; TraAddress is where the package goes to; CCtype is which kind of Credit Card type this order used in the progressing.**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Description** |
| OrderNum | CHAR | Order number of the buying action |
| CustomerID | INTEGER | Foreign key to CUSTOMER |
| ISBN | CHAR | Foreign key to BOOK |
| Quantity | INTEGER | The quantity of the book the order bought |
| Amount | INTEGER | Total price of this order |
| BDate | INTEGER | Order date |
| TraAddress | VARCHAR | Address the package goes |
| CCType | VARCHAR | Credit card type when checking |

### **B. The sample SQL queries** These queries should be organized and presented neatly and professionally. Each query should include: An English language description of what the query should be returning The correct relational algebra syntax of the query The equivalent SQL query

#### **CheckPoint 02 normal Query**

##### **2a. "Find all Book Titles written by Ann Patchett with a price less than $10"**

**SELECT B.Title**

**FROM BOOK AS B, WRITTENBY AS W, AUTHOR AS A**

**WHERE B.ISBN = W.ISBN**

**AND W.AuthorID = A.AuthorID**

**AND B.Price < 10**

**AND A.LName = 'Pratchett' AND A.FName='Ann';**

##### **2b. Give all the titles and their dates of purchase made by a single customer**

**SELECT B.Title, T.BDate**

**FROM BOOK AS B, TRANSACTS AS T, CUSTOMER AS C**

**WHERE C.CustomerID = 1111111111**

**AND T.CustomerID = C.CustomerID**

**AND B.ISBN = T.ISBN;**

##### **2c. Find the titles and ISBNs for all books with less than 5 copies in stock**

**SELECT B.Title, B.ISBN**

**FROM BOOK AS B, STORES AS S**

**WHERE StockNum < 5**

**AND B.ISBN = S.ISBN;**

**2d. Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased**

**SELECT DISTINCT C.CustomerID, C.CFName, C.CLName, B.Title**

**FROM BOOK AS B, WRITTENBY AS W, AUTHOR AS A, CUSTOMER AS C, TRANSACTS AS T**

**WHERE B.ISBN = W.ISBN**

**AND W.AuthorID = A.AuthorID**

**AND T.CustomerID = C.CustomerID**

**AND T.ISBN = B.ISBN**

**AND A.LName = 'Pratchett'**

**ORDER BY C.CustomerID;**

##### 

##### **2e. Find the total number of books purchased by a single customer**

**SELECT C.CFName, C.CLName, SUM(T.Quantity)**

**FROM CUSTOMER AS C, TRANSACTS AS T**

**WHERE T.CustomerID = C.CustomerID**

**AND C.CustomerID = 1111111111**

**GROUP BY C.CustomerID;**

##### **2f. Find the customer who has purchased the most books and the total number of books they have purchased**

**SELECT Count\_Books.CustomerID,Count\_Books.CFName, Count\_Books.CLName, MAX(S)**

**FROM (SELECT C.CustomerID, C.CFname, C.CLname, SUM(Quantity) as S**

**FROM CUSTOMER AS C, TRANSACTS AS T**

**WHERE T.CustomerID = C.CustomerID**

**GROUP BY C.CustomerID) as Count\_Books;**

#### **CheckPoint 02 extra Query**

##### **Extra1: Show customers names and its contact phone number who have bought the amount of the book greater than 3**

**SELECT Count\_Books.CustomerID,Count\_Books.CFName, Count\_Books.CLName, Count\_Books.S, Count\_Books.CPhNum**

**FROM (SELECT C.CustomerID, C.CFName, C.CLName,C.CPhNum,SUM(Quantity) as S**

**FROM CUSTOMER AS C, TRANSACTS AS T**

**WHERE T.CustomerID = C.CustomerID**

**GROUP BY C.CustomerID) as Count\_Books**

**WHERE Count\_Books.S > 3;**

##### **Extra1: Another Join commands**

**SELECT Count\_Books.CustomerID,Count\_Books.CFName, Count\_Books.CLName, Count\_Books.S, Count\_Books.CPhNum**

**FROM(SELECT C.CustomerID,C.CFname,C.CLname,C.CPhNum,SUM(Quantity) as S**

**FROM CUSTOMER AS C**

**JOIN TRANSACTS AS T**

**ON T.CustomerID = C.CustomerID**

**GROUP BY C.CustomerID) as Count\_Books**

**WHERE Count\_Books.S > 3;**

##### **Extra2: Show publishers names who have published at least two books stored on my bookstore whose price of each book is greater than 40 by B.Price**

**SELECT P.PName, COUNT(P.PubID) AS PubTotalTypeB**

**FROM BOOK B, PUBLISHER P**

**WHERE B.PubID = P.PubID AND B.Price > 40**

**GROUP BY P.PubID**

**HAVING COUNT(P.PubID)>= 2;**

##### **Extra2: Another Join commands**

**SELECT P.PName, COUNT(P.PubID) AS PubTotalTypeB**

**FROM BOOK AS B**

**JOIN PUBLISHER AS P ON B.PubID = P.PubID**

**WHERE B.Price > 40**

**GROUP BY P.PubID**

**HAVING COUNT(P.PubID)>= 2;**

##### **Extra3: Show publishers names who have published books whose price is greater than 40 and the stock is greater than 80**

**SELECT P.PName, P.PubID, Count\_Stock.TotalStock, B.ISBN, B.Title, B.Price**

**FROM BOOK B,PUBLISHER P, (SELECT S.ISBN, SUM(S.StockNum) AS TotalStock**

**FROM STORES S**

**GROUP BY S.ISBN) AS Count\_Stock**

**WHERE B.PubID = P.PubID AND Count\_Stock.ISBN = B.ISBN AND Price > 40 AND Count\_Stock.TotalStock > 40**

**GROUP BY B.ISBN**

**ORDER BY P.PubID, B.Price;**

##### **Extra3: Another Join commands**

**SELECT P.PName, P.PubID, Count\_Stock.TotalStock, B.ISBN, B.Title, B.Price**

**FROM BOOK B**

**JOIN PUBLISHER P ON B.PubID = P.PubID**

**JOIN (SELECT S.ISBN, SUM(S.StockNum) AS TotalStock**

**FROM STORES S**

**GROUP BY S.ISBN) Count\_Stock ON Count\_Stock.ISBN = B.ISBN**

**WHERE Price > 40 AND Count\_Stock.TotalStock > 40**

**GROUP BY B.ISBN**

**ORDER BY P.PubID, B.Price;**

#### **CheckPoint 03: Advanced query about the additional schema**

##### **Advanc(a).Provide a list of customer names, along with the total dollar amount each customer has spent.**

**SELECT C.CustomerID,C.CFName,C.CLName, SUM(T.Amount) AS TotalConsume**

**from CUSTOMER C, TRANSACTS T**

**WHERE C.CustomerID = T.CustomerID**

**GROUP BY C.CustomerID;**

##### **Advanc(b).Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.**

**SELECT C.CFName,C.CLName, C.Email, C.CustomerID**

**from CUSTOMER C**

**JOIN TRANSACTS T ON C.CustomerID = T.CustomerID**

**GROUP BY C.CustomerID**

**HAVING SUM(T.Amount) > (SELECT AVG(Total.TotalConsume)**

**from (SELECT SUM(T.Amount) AS TotalConsume**

**FROM CUSTOMER C, TRANSACTS T**

**WHERE C.CustomerID = T.CustomerID**

**GROUP BY C.CustomerID) AS Total);**

##### **Advanc(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 Title, SUM(T.Quantity) AS AmountSold\_Quantity**

**FROM BOOK B**

**JOIN TRANSACTS T**

**ON B.ISBN = T.ISBN**

**GROUP BY B.ISBN**

**ORDER BY AmountSold\_Quantity DESC;**

##### **Advanc(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 Title, SUM(T.AMOUNT) AS AmountSold\_Copies**

**FROM BOOK B**

**JOIN TRANSACTS T**

**ON B.ISBN = T.ISBN**

**GROUP BY B.ISBN**

**ORDER BY AmountSold\_Copies DESC;**

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

**SELECT Author\_sold.AuthorID, A.FName, A.LName, A.MName, Author\_sold.AmountlSold**

**FROM AUTHOR A, (SELECT SUM(T.Quantity) AS AmountlSold, A.AuthorID**

**from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID AND B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS Author\_sold**

**WHERE A.AuthorID = Author\_sold.AuthorID AND Author\_sold.AmountlSold**

**=(SELECT MAX(Author\_sold.AmountlSold)**

**FROM (SELECT SUM(T.Quantity) AS AmountlSold, A.AuthorID**

**from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID AND**

**B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS Author\_sold);**

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### **Advanc(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\_sold.AuthorID, A.FName, A.LName, A.MName, Author\_sold.AmountlSold**

**FROM AUTHOR A, (SELECT SUM(T.Amount) AS AmountlSold, A.AuthorID**

**from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID**

**AND B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS Author\_sold**

**WHERE A.AuthorID = Author\_sold.AuthorID AND Author\_sold.AmountlSold**

**=(SELECT MAX(Author\_sold.AmountlSold)**

**FROM (SELECT SUM(T.Amount) AS AmountlSold, A.AuthorID**

**from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID AND**

**B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS Author\_sold);**

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

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

**SELECT distinct C.CustomerID, C.CFName, C.CLName, C.CPhNum**

**from BOOK B, AUTHOR A, WRITTENBY W,TRANSACTS T, CUSTOMER C**

**WHERE B.ISBN = W.ISBN AND A.AuthorID = W.AuthorID AND B.ISBN= T.ISBN AND C.CustomerID= T.CustomerID**

**AND A.AuthorID=(SELECT Author\_sold.AuthorID**

**FROM AUTHOR A, (SELECT SUM(T.Amount) AS AmountlSold,**

**A.AuthorID**

**FROM BOOK B, AUTHOR A, WRITTENBY**

**W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN AND A.AuthorID =**

**W.AuthorID AND B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS Author\_sold**

**WHERE A.AuthorID = Author\_sold.AuthorID AND**

**Author\_sold.AmountlSold =(SELECT**

**MAX(Author\_sold.AmountlSold)**

**FROM (SELECT SUM(T.Amount) AS**

**AmountlSold, A.AuthorID**

**FROM BOOK B, AUTHOR A,**

**WRITTENBY W,TRANSACTS T**

**WHERE B.ISBN = W.ISBN**

**AND A.AuthorID = W.AuthorID**

**AND B.ISBN= T.ISBN**

**GROUP BY A.AuthorID) AS**

**Author\_sold) )**

**ORDER BY C.CustomerID;**

##### **Advanc(h).Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.**

**SELECT DISTINCT A.AuthorID, A.FName, A.LName**

**from CUSTOMER C**

**JOIN TRANSACTS T ON C.CustomerID = T.CustomerID**

**JOIN BOOK B ON T.ISBN = B.ISBN**

**JOIN WRITTENBY W ON W.ISBN = B.ISBN**

**JOIN AUTHOR A ON A.AuthorID= W.AuthorID**

**WHERE C.CustomerID IN (SELECT C.CustomerID**

**FROM CUSTOMER C**

**JOIN TRANSACTS T ON C.CustomerID = T.CustomerID**

**GROUP BY C.CustomerID**

**HAVING SUM(T.Amount) >**

**(SELECT AVG(Total.TotalConsume)**

**FROM (SELECT SUM(T.Amount) AS TotalConsume**

**FROM CUSTOMER C, TRANSACTS T**

**WHERE C.CustomerID = T.CustomerID**

**GROUP BY C.CustomerID) AS Total) );**

#### **Relational Algebra:**

2a.

RA: BOOK ⨝ (ISBN =ISBN) WRITTENBY

RB: RA ⨝ (RA.AuthorID = AUTHOR.AuthorID AND price<10 AND

LName =‘Pratchett’ AND FNmae=’ANN’) AUTHOR

RESULT: π(Title) RB

2b.

RA: (CUSTOMER \* ISBN \*TRANSACTION)

RB: σ(CustomerID=1111111111) RA

RESULT: π(Title,Bdate) RB

**2c.**

RA: BOOK X STORES

RB: σ(StockNum<5 AND BOOK.ISBN = STORES.ISBN) RA

RESULT: π(Title, ISBN) RB

##### **2d**

R1: CUSTOMER⨝ (CUSTOMER.CustomerID=TRANSACTS.CustomerID)TRAN

STACTS

R2: R1⨝ (R1.ISBN=BOOK.ISBN)BOOK

R3: R2⨝ (R2.ISBN=WRITTENBY.ISBN)WRITTENBY

R4: R3⨝ (R3.AuthorID=AUTHOR.AuthorID)AUTHOR

R5: σ(LName = 'Pratchett') R4

RESULT: π(distinct(CustomerID,CFName,CLName,Title) R5

**2e.**

RA: TRANSACS ⨝ (CustomerID = CustomerID AND CLName = ‘WENG’)

CUSTOMER

RB: CUSTOMER.CustomerID F (SUM(TRANSACS.Quantity))RA

RC:CUSTOMER X RB

RESULT: π(FName, LName, Sum\_Quantity)RC

##### **2f.**

TEMP: CUSTOMER⨝(CUSTOMER.CustomerID=TRANSACTS.CustomerID)T

RANSTACTS

Count\_Books: πCustomerID,CFName,CLName,Quantity(TEMP)

RESUL: πCustomerID,CFName,CLName,MAX(Quantity)(Count\_Books)

##### **Extra01**:

R1A: σ( (CustomerID Fas TotalQuan) >3 ) (CUSTOMER \* TRANSACTS)

Result: π( CustomerID,CFName,.CLName, totalQuan, CPhNum )R1A

##### **Extra02:**

R1A: σ( Price > 40 )BOOK

R1B:σ( (PubID Fas TotalPub) >= 2 ) PUBLISHER⨝PubID=PubID R1A

Result: π(PName, TotalPub ) R1B

##### **Extra03:**

R1A: σ( Price > 40 )BOOK

R1B:σ( (PubID Fas TotalSto) >= 80 ) PUBLISHER⨝PubID=PubID R1A

Result: π(PName, TotalSto ) R1B

##### **Advanc(a):**

R1A: CUSTOMER⨝CustomerID=CustomerIDTRANSACTS

Result: π(CFName, CLName, CustomerIDF ) R1A

##### **Advanc(b):**

**R1A: σ(CustomerIDFCUSTOMER⨝CustomerID=CustomerIDTRANSACTS**

**R1B: CustomerIDF) R1A**

**R1C: σ(SUM(T.Amount) > R1B) CUSTOMER⨝CustomerID=CustomerIDTRANSACTS**

**Result: π(CFName,CLName, Email ) R1C**

##### **Advanc(c):**

R1A:σISBNF) TRANSECTION

Result: π(Title)BOOK⨝ISBN=ISBN R1A

##### **Advanc(d):**

RA : σ( (ISBN Fas AmountlSold\_Copies) ) (BOOK ⨝ (ISBN = ISBN) TRANSACTS))

Result: π(Title, AmountSold\_Copies) (RA \* BOOK)

##### **Advanc(e):**

R1A: σ( AuthorID, (AuthorID Fas AmountlSold) ) (BOOK \* AUTHOR \* WRITTENBY\* TRANSACTS)

R1B: (R1A ⨝ AuthorID=AuthorID AUTHOR)

R1C: σ( AuthorID, (AuthorID Fas AmountlSold) )  (BOOK \* AUTHOR \* WRITTENBY \* TRANSACTS)

R1D: π(AuthorID F) (R1C)

Result: (R1B ⨝ AmountlSold=MaxSold R1D)

##### **Advanc(f)**

R1A: σ( AuthorID, (AuthorID Fas AmountlSold) )  (BOOK \* AUTHOR \* WRITTENBY \* TRANSACTS)

R1B: (R1A ⨝ AuthorID=AuthorID AUTHOR)

R1C: σ( AuthorID, (AuthorID Fas AmountlSold) )  (BOOK \* AUTHOR \* WRITTENBY \* TRANSACTS)

R1D: π(AuthorID F) (R1C)

ResultF: (R1B ⨝ AmountlSold=MaxSold R1D)

##### **Advanc(g)**

RA: CUSTOMERS ⨝ (CusomerID = CustomerID) TRANSACTS

RB:RA⨝ (ISBN = ISBN)BOOK

RC: RB⨝(AuthorID=AuthorID) WRITTENBY

RD: RC⨝ (AuthorID = AuthorID)ResultF

RESULTG: π(CustomerID,CFName,CLName,CPhNum)RD

##### **Advanc(h)**

R1: σDISTINCT(AuthorID,FName,LName)(CUSTOMER)⋈CUSTOMER.CustomerID = TRANSTACT.CustomerIDσTRANSACTS

R2: R1⋈R1.ISBN = BOOK.ISBN(BOOK)

R3: R2⋈R2.ISBN = WRINTTENBY.ISBN(WRITTENBY)

R4: R3⋈R3.AuthorID= AUTHOR .AuthorID(AUTHOR)

R5: CUSTOMER⋈(CUSTOMER.CustomerID=TRANSACTS.CustomerID)TRANSTACT

R6: πSUM(R1.Amount)

R7: πAVG(R2)

R8: πCustomerID(R5)σ(SUM(Amount)>R7)

RESULT: πAuthorID,FName,LName(R8)

### C. Syntax for: INSERT syntax for adding new books, publishers, authors and customers to your system. If there are dependencies in your system that require multiple records to be added to tables in a specific order to add one of these items, make sure you clearly indicate what those restrictions are. DELETE syntax for removing books, publishers, authors and, customers from your system. Again, indicate any dependencies that exist on the order that the steps in your DELETE must take. In addition, provide an example set of DELETE statements for each entity in your database.

#### **1. The Insert/Delete instruction of BOOK table**



(The original table of BOOK information)

INSERT SYNTAX TEST:

Example: Calling the following command to insert books:

INSERT INTO BOOKSTORE.BOOK(ISBN, PubID, Price, Title, BKLanguage) values ("0000000016",0000005,168.99,"Chinese History","Chinese");

INSERT INTO BOOKSTORE.BOOK(ISBN, PubID, Price, Title, BKLanguage) values ("0000000017",0000007,32.99,"What I believe","English");



(Result after calling INSERT)

DELETE SYNTAX TEST:

\*If we want to delete a book in the database,

\*we need to delete the book step by step according to the relation model design

Example: Calling the following command to delete a book with ISBN, “0000000010”:

DELETE FROM BOOKSTORE.STORES

WHERE ISBN="0000000010";

DELETE FROM BOOKSTORE.TRANSECTION

WHERE ISBN="0000000010";

DELETE FROM BOOKSTORE.WRITTENBY

WHERE ISBN="0000000010";

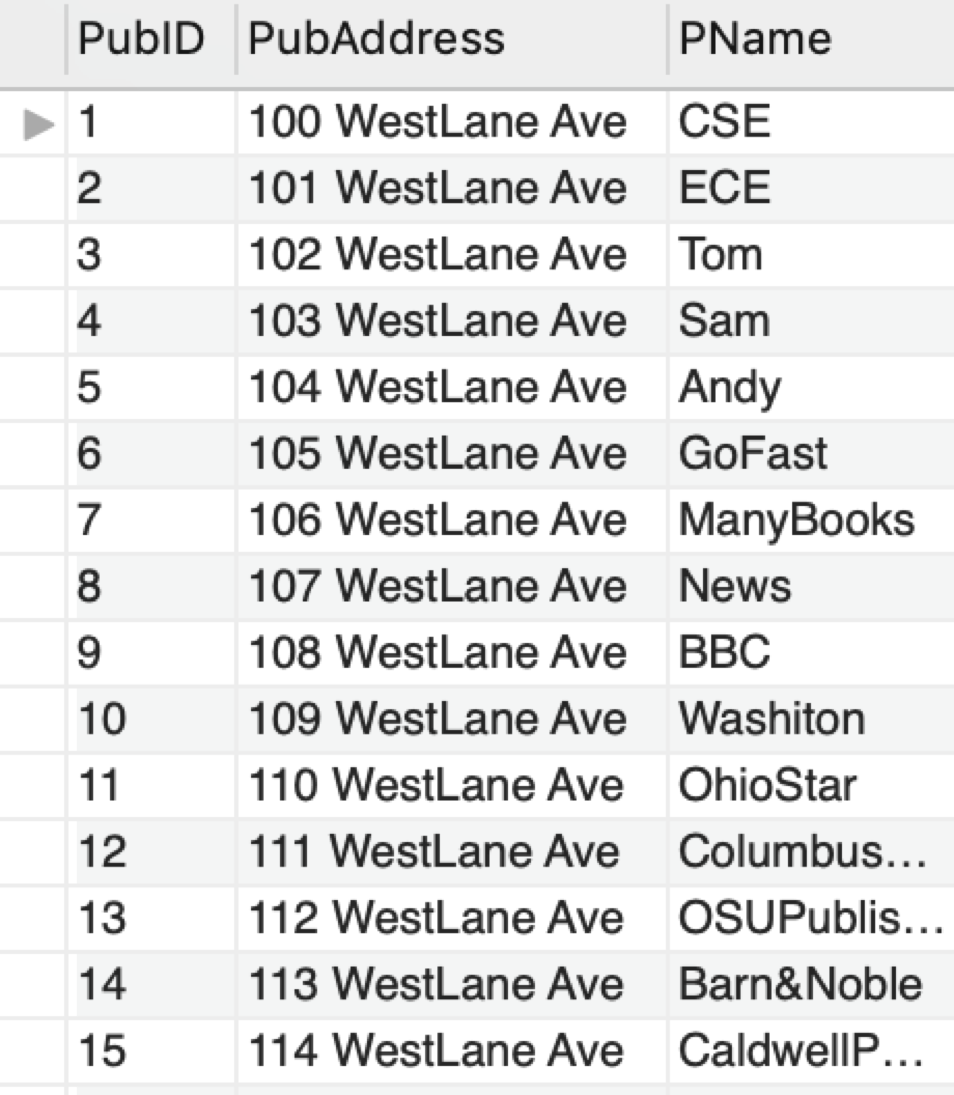
DELETE FROM BOOKSTORE.BOOK

WHERE ISBN="0000000010";



(Result after calling DELETE)

#### **2. The Insert/Delete instruction of PUBLISHER table**

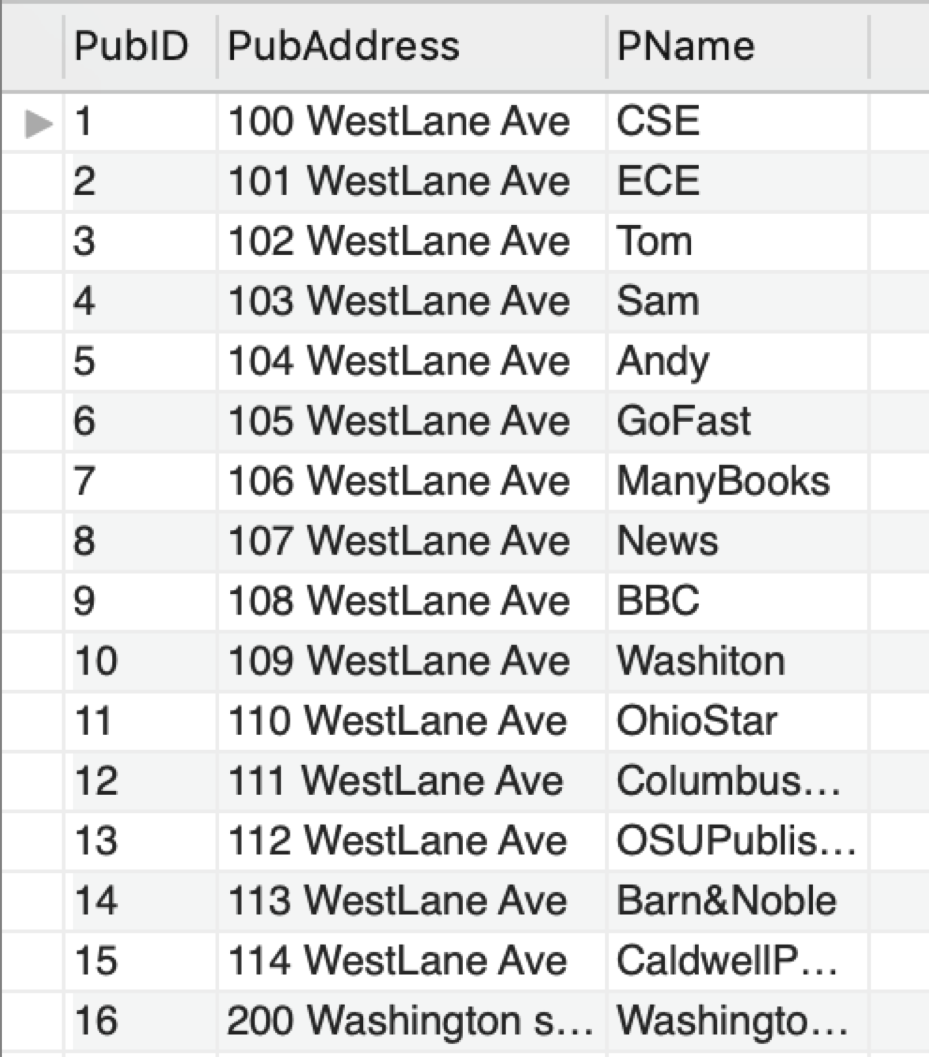


(The original table of PUBLISHER information)

INSERT SYNTAX TEST:

Example: Calling the following command to insert a publisher:

INSERT INTO BOOKSTORE.PUBLISHER(PubID, PubAddress, PName) values (00000016,"200 Washington street","Washington Post");

****

(Result after calling INSERT)

DELETE SYNTAX TEST:

\*If we would like to delete a publisher in the database,

\*we need to delete any relationship with the primary key,PubID, PUBLISHER has.

Example: Calling the following command to delete a publisher with PubID, 00000010.

DELETE FROM BOOKSTORE.STORES

WHERE ISBN IN (SELECT B.ISBN

FROM BOOK B, PUBLISHER P

WHERE B.PubID = P.PubID

AND P.PubID =00000010);

DELETE FROM BOOKSTORE.TRANSECTION

WHERE ISBN IN (SELECT B.ISBN

FROM BOOK B, PUBLISHER P

WHERE B.PubID = P.PubID

AND P.PubID =00000010);

DELETE FROM BOOKSTORE.WRITTENBY

WHERE ISBN IN (SELECT B.ISBN

FROM BOOK B, PUBLISHER P

WHERE B.PubID = P.PubID

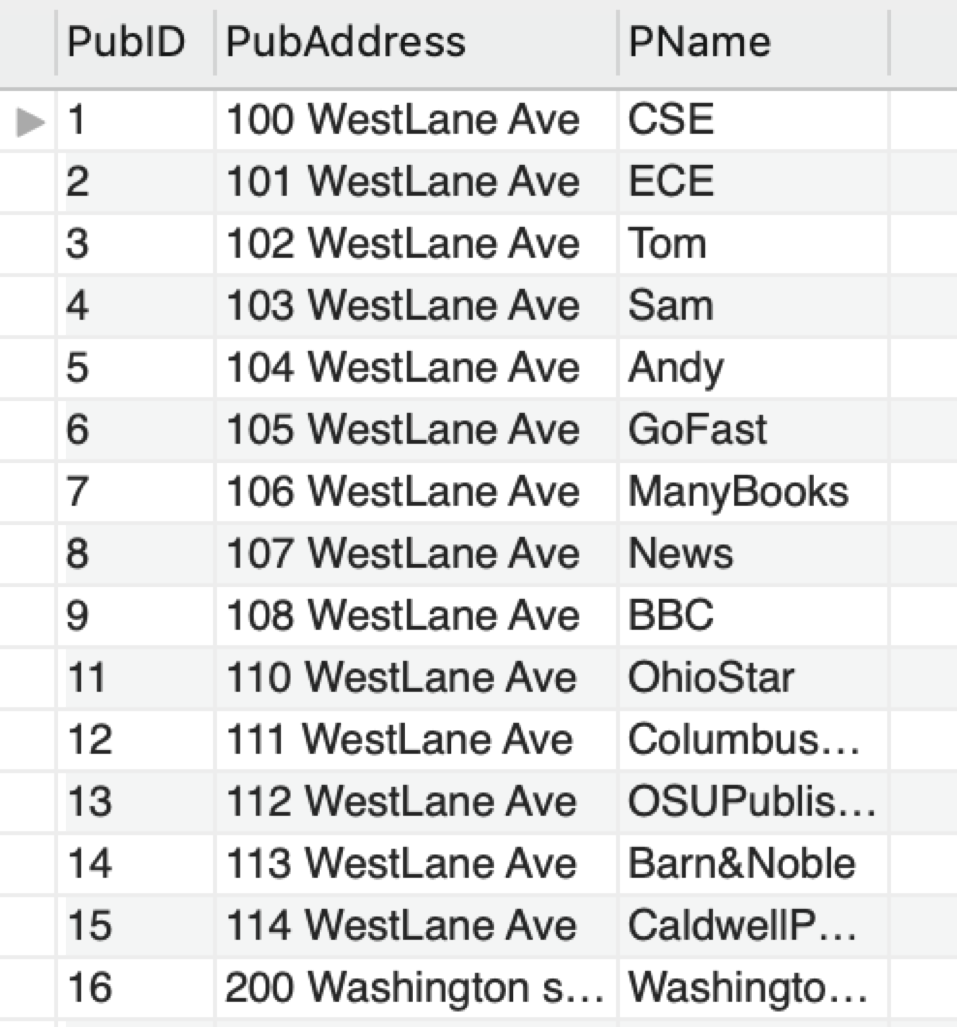
AND P.PubID =00000010);

DELETE FROM BOOKSTORE.BOOK

WHERE PubID =00000010;

DELETE FROM BOOKSTORE.PUBLISHER

WHERE PubID =00000010;

****

(Result after calling DELETE)

#### **3. The Insert/Delete instruction of AUTHOR table**

****

(The original table of AUTHOR information)

##### 

##### INSERT SYNTAX TEST:

Example: Calling the following command to insert an author:

INSERT INTO BOOKSTORE.AUTHOR(AuthorID,Fname, Lname, Mname) values (00016,"LANCE","CHEN"," ");

****

(Result after calling INSERT)

##### DELETE SYNTAX TEST:

\*If we would like to delete an author in the database,

\*we need to delete any relationship with the primary key,AuthorID, AUTHOR has.

Example: Calling the following command to delete an author with AuthorID, 0010.

DELETE FROM BOOKSTORE.WRITTENBY

WHERE AuthorID=0010;

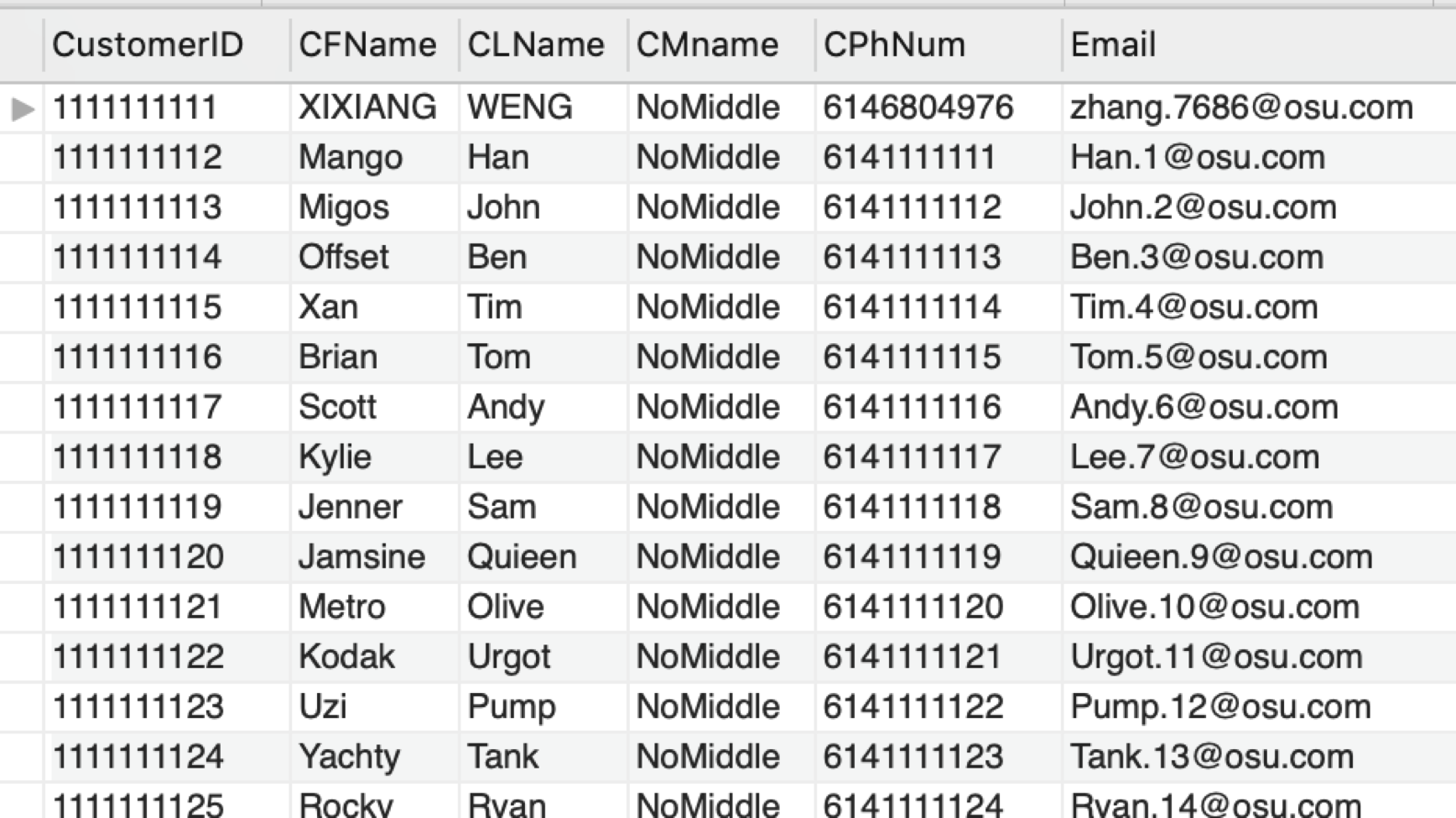
DELETE FROM BOOKSTORE.AUTHOR

WHERE AuthorID=0010;

****

(Result after calling DELETE)

#### **4. The Insert/Delete instruction of CUSTOMER table**

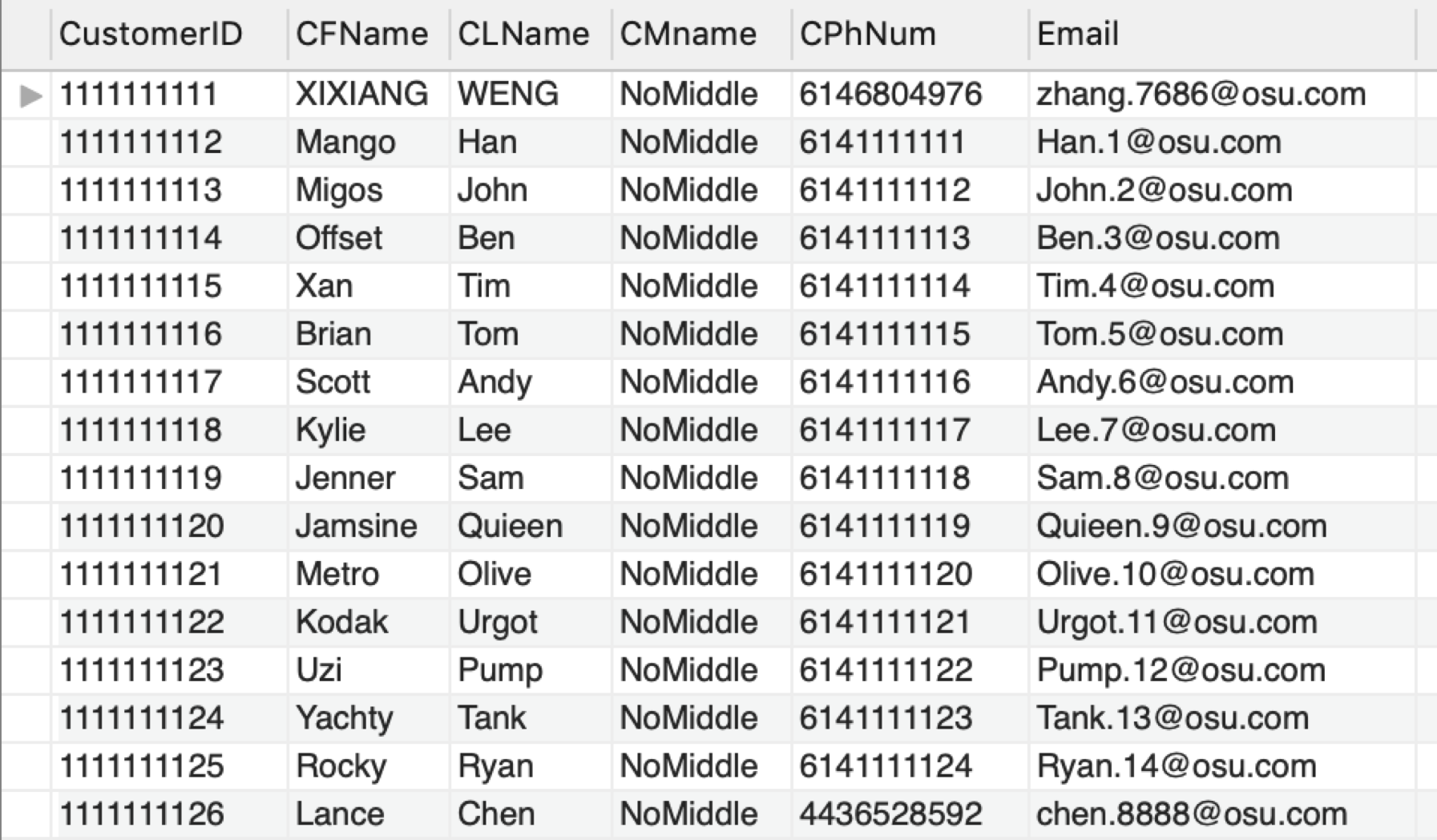
****

(The original table of CUSTOMER information)

INSERT SYNTAX TEST:

Example: Calling the following command to insert a customer information:

INSERT INTO BOOKSTORE.CUSTOMER(CustomerID,CFname,CLname,CPhNum, Email) values (1111111126,"Lance","Chen","4436528592","chen.8888@osu.com");

****

(Result after calling INSERT)

##### DELETE SYNTAX TEST:

\*If we want to delete a customer in the database,

\*we need to delete the customer step by step by its relationship.

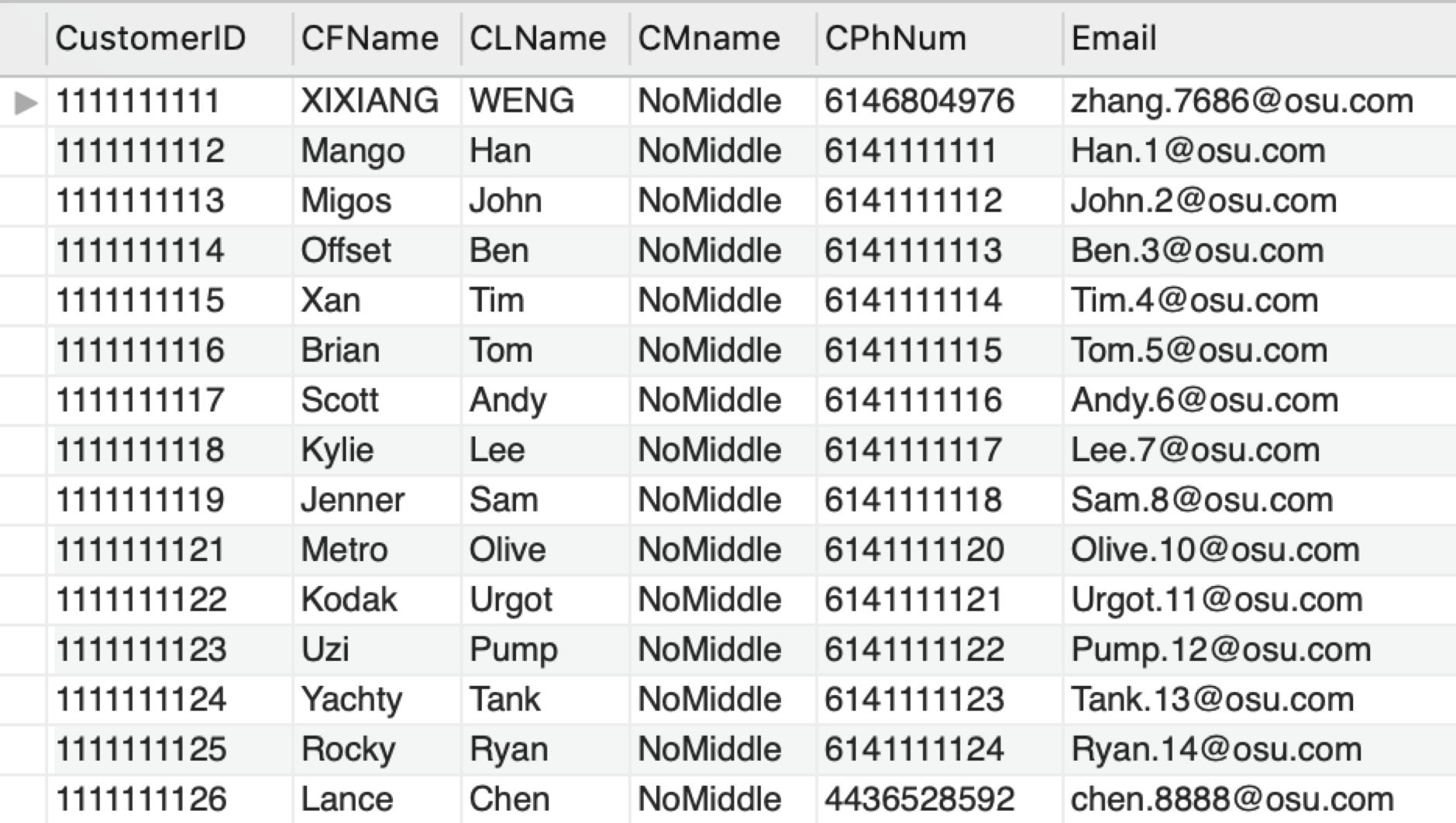
Example: Calling the following command to delete a customer with CustomerID, “1111111120”:

DELETE FROM BOOKSTORE.TRANSECTION

WHERE CustomerID=1111111120;

DELETE FROM BOOKSTORE.CUSTOMER

WHERE CustomerID=1111111120;

****

(Result after calling DELETE)

## **SECTION 3:Graded checkpoints documents:**

## An appendix to the final report that MUST contain all your original, graded checkpoint documents organized in a neat and professional manner. For each checkpoint that required a revision, you MUST include a revision for that checkpoint. This revision may be a pointer to where in the final database document the “fixed” version of the checkpoint resides (“See Section X Page Y for the new relational model diagram” for example).

#### CHECKPOINT 01:

**Revised the relational schema and ER diagram**

**See Section 3. Page 34: "Revised ER diagram for Check01”.**

**See Section 3. Page 36: "Revised Relational Schema for Check01"**

#### CHECKPOINT 02:

**Revised the relational schema and ER diagram, also revised the relational algebra of the whole database**

**See Section 3. Page 35: “Revised ER diagram for Check02”.**

**See Section 3 Page 37: "Revised Relational Schema for Check02"**

#### CHECKPOINT 03:

**Revised the relational schema and ER diagram and give hints about the notation.**

**And revised the code logic for query**

**See Section 3. Page 36: “Revised ER diagram for Check03”.**

**See Section 3. Page 37: "Revised Relational Schema for Check03"**

**See Section 2. Page 13: "B. Sample SQL queries"**

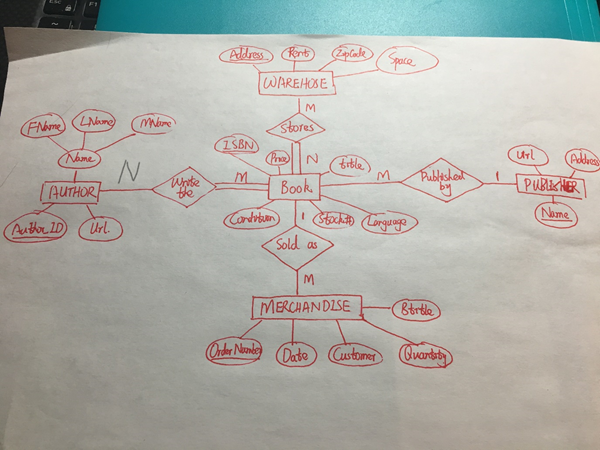
#### CHECKPOINT 04:

**Revised the relational schema and ER diagram. Besides, correct the grammar of creating a view**

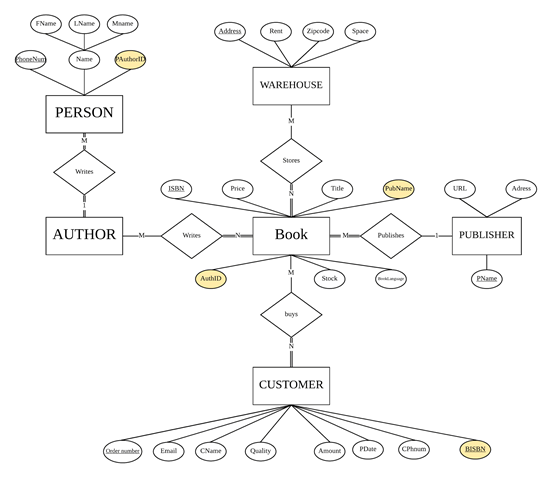
**See Section 1. Page 2: "Database description" to see the changes in ER diagram.**

**See Section 1. Page 2: "Relational Schema" to see the changes of the schema.**

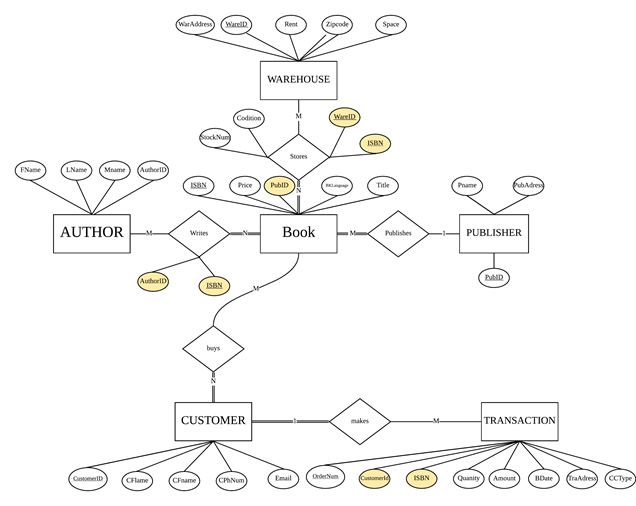
**See Section 1. Page 7: "View Description" to see the changes of view operation.**

****

**Figure 01: "Revised ER diagram for Check01”.**

****

**Figure 02: "Revised ER diagram for Check02”.**

****

**Figure 03: "Revised ER diagram for Check03”.**

**• AUTHOR: Fname, Lname, Mname, ID  
• WAREHOUSE: Address, Rent, ZipCode, Space  
• BOOK: ISBN, Price, Title, Condition, Stock#, Language, AuthID, PubName  
• PUBLISHER: Name, Url, Address  
• MERCHANDISE: OrderNum, Quantity, Customer, BTitle, ISBN\_Num  
• AUTHRO\_BOOK: ISBN\_Num, AuthID  
• WAREHOUSE\_BOOK:ISBN\_Num,WareAddress**

**Table01: "Revised Relational Schema for Check01"**

#### ● **AUTHOR**: AuthorID

#### ● **PERSON:** Fname, Lname, Mname, PhoneNum, **PAuthorID**

#### ● **WAREHOUSE:** Address, Rent, ZipCode, Space

#### ● **BOOK**: ISBN, Price, Title, Condition, Stock,BookLanguage, **PubName, AuthID**

#### ● **PUBLISHER**: PName, URL, Address

#### ● **CUSTOMER:** OrderNum,CName,CPhNum,Quantity,Amount, PDate,Email, **BISBN**

#### ● **BOOK\_WrittenBy\_AUTHOR: BKISBN, AuID**

#### ● **CUSTOMER\_BUYS\_BOOK: BookISBN, CusOrderNum**

#### ● **WAREHOUSE\_STORES\_BOOK: BoISBN ,Wadresss**

**Table02: "Revised Relational Schema for Check02"**

* **AUTHOR: AuthorID , Fname, Lname, Mname**
* **BOOK: ISBN, PubID, Price, Title, BKLanguage**
* **WRITTENBY: ISBN, AuthorID (Note: The red color represents the foreign key)**
* **PUBLISHER: PubID, PubAddress, PName**
* **WAREHOUSE: WareID, WarAddress, Rent, ZipCode, Space**
* **STORES: ISBN ,WareID, BKCondition, StockNum**
* **CUSTOMER: CustomerID,CFname,CLname,CPhNum, Email**
* **TRANSACTION: OrderNum, CustomerID, ISBN, Quantity, Amount, BDate, TraAddress, CCType**

**Table03: "Revised Relational Schema for Check03"**