# USER'S MANUAL

*Bits and Books Inventory and Sales Database Suite*

Prepared by: Chen Cai, Tingwei Duan, Huei Li, En-Ju Lin

June, 2016

## Revision Sheet

| Release No. | Date | Revision Description |
|---|---|---|
| Rev. 0 | 6/24/16 | User's Manual Created |
| | | |
| | | |
| | | |

# 1.0 SYSTEM SUMMARY

# 1.0 GENERAL INFORMATION

## 1.1 System Overview

The Inventory & Sales Database Suite (ISDS) is a relational database management system based on SQLite. The ISDS provides a graphical user interface that allows a user to input all information and store it electronically to exterminate the need for using paper records. This database supports the inventory management and sales operation of Bits and Books.

## 1.2 Abbreviations

PK = Primary Key
UK = Unique Key
FK = Foreign Key
ISBN = International Standard Book Number

## 1.3 Database Relation Description

The database contains 19 relations as described in the tables below.

### PUBLISHER

Contains information on the primary email address and mailing address of publishers whose books we sell or has sold in the past but no longer carry.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **P_Name** | Variable string of up to 50 characters | PK | Must enter value | Publisher Name |
| **P_Email** | Variable string of up to 50 characters | UK | Must enter value | Publishers' primary email address for contact. Only one allowed for each publisher |
| **P_Address** | Variable string of up to 50 characters | FK references to ADDRESS (Address) for current US addresses | Must enter value.<br><br>A subset of ADDRESS (Address) domain<br><br>On reference delete - Set Null<br><br>On reference update - Cascade | Publisher's primary mailing address. Only one allowed for each publisher. Enter only street and street number |
| **P_ZIP** | A fixed string of 5 characters | FK references to ADDRESS (ZIP) to derive City and State information | Must enter value.<br><br>A subset of ADDRESS (ZIP) domain<br><br>On reference delete - Set null<br><br>On reference update - Cascade | Publisher's primary mailing address zip code. This support the 5 character zip code format for US address |

## PUBLISHER_PHONE

Contains information on the phone number(s) of publishers whose books we sell or has sold in the past but no longer carry. There can be zero to many phone numbers associated with a publisher.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **P_Name** | Variable string of up to 50 characters | PK,<br>FK references to PUBLISHER (P_Name) | Must enter value<br><br>A subset of PUBLISHER (P_Name) domain<br><br>On reference delete - Set null<br><br>On reference update - Cascade | Publisher Name |
| **P_Phone** | A fixed string of 12 characters in the form of 123-456-7890 | | Must enter value | Publishers' phone numbers. One phone number per cell but a publisher can store more than one phone numbers |

## BOOK

Contains information on the books we sell or had sold. .

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK | Must enter value | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **Title** | Variable string of up to 150 characters | | Must enter value | Book title |
| **Year** | Date variable | | Must enter value | Year book published |
| **Format** | Variable string of up to 10 characters | | Must enter value | Type of print – Hardcover, Paperback, Kindle or Audio |
| **Language** | Variable string of up to 20 characters | | Must enter value | Language book written in |
| **Price** | Number to 2 decimal place, up to 6 digits | | Must enter value | Book price |

## CATEGORY_KEY

Provide the key to each category according to the major headings of the BISAC Subject Heading List.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **Category_ID** | Variable string of up to 4 characters | PK | Must enter value | Unique ID of 4 alphanumerical characters assigned to each book category. |
| **Category** | Variable string of up to 30 characters | | Must enter value | Categorization of books using the BISAC major headings, an industry standard. For the list of categories, refer to https://www.bisg.org/bisac/complete-bisac-subject-headings-2015-edition |

## CATEGORY

Provide category information for each book using Category_ID.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **Category_ID** | Variable string of up to 4 characters | PK<br>FK references to CATEGORY_KEY (Category_ID) | Must enter value<br><br>A subset of CATEGORY_ID (Category_ID) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | Links to unique Category ID |

## BOOK_PUBLISHER

Connects book with publisher information

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **P_Name** | Variable string of up to 50 characters | PK<br>FK references to PUBLISHER (P_Name) | Must enter value<br><br>A subset of PUBLISHER (P_Name) domain<br><br>On reference delete - Set null<br><br>On reference update - Cascade | Publisher Name |

## CUSTOMER_INFO

Contains customer information, with email as unique identification for individual customer.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **C_Email** | Variable string of up to 50 characters | PK | Must enter value | Customer registered email for unique user account. |
| **C_FName** | Variable string of up to 20 characters | | | Customer's first name |
| **C_LName** | Variable string of up to 20 characters | | | Customer's last name |
| **C_Phone** | A fixed string of 12 characters in the form of 123-456-7890 | | | Customer's phone number |

## CUSTOMER_PAYMENT

Customers can optionally choose to save one or more credit card information linked to their account.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **C_Email** | Variable string of up to 50 characters | PK<br>FK references to CUSTOMER_INFO (C_Email) | Must enter value<br><br>A subset of CUSTOMER_INFO (C_Email) domain<br><br>On reference delete<br>- Cascade<br><br>On reference update<br>- Cascade | Customer registered email for unique user account |
| **C_Card** | Variable string of up to 50 characters | | | Customer's credit card information, including the card number, CVV number and expiry date This is field is encrypted |

## ADDRESS

Data pulled from US address database, including street name and zip code.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **Address** | Variable string of up to 50 characters | PK | Must enter value | Valid US Street address |
| **ZIP** | A fixed string of 5 characters | PK<br>FK references to ZIPCODE (ZIP) to derive City and State information | Must enter value | Valid ZIP code associated with the street address |

## ZIPCODE

Data pulled from US address database, containing all US 5 digit Zip Code with corresponding city and state.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ZIP** | A fixed string of 5 characters | PK | Must enter value | Valid US Zip Code |
| **City** | Variable string of up to 30 characters | | Must enter value | Valid city name associated with the Zip Code |
| **State** | A fixed string of 2 characters | | Must enter value | 2 letter abbreviation of State Name. e.g OH for Ohio |

## ORDERS

Contains customer order transaction information. Each order transaction is assigned an unique Order_ID.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ORDER_ID** | Variable string of up to 10 characters | PK | Must enter value | Unique Order_ID to identify each transaction |
| **Date** | Date in the form of YYYY-MM-DD | | Must enter value | Date of order transaction submission |
| **Status** | Variable string of up to 10 characters | | Must enter value | Status of order: Cancelled, Pending or Completed |
| **PaymentInfo** | Variable string of up to 50 characters | | Must enter value | Payment detail relating to the transaction such as credit card information |
| **C_Email** | Variable string of up to 50 characters | FK references to CUSTOMER (C_Email) | A subset of CUSTOMER_INFO (C_Email) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | Customer who made the order |

## WAREHOUSE

Contains information on the warehouses, each with unique identifier.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **W_Code** | A fixed string of 5 characters | PK | Must enter value | Unique identifier for different warehouses |
| **W_Phone** | A fixed string of 12 characters in the form of 123-456-7890 | UK | Must enter value | Warehouse's primary phone for contact |
| **W_Address** | Variable string of up to 50 characters | FK references to ADDRESS (Address) for current US addresses | Must enter value.<br><br>A subset of ADDRESS (Address) domain<br><br>On reference delete - Set Null<br><br>On reference update - Cascade | Warehouse address, include street number and street name |
| **W_ZIP** | A fixed string of 5 characters | FK references to ADDRESS (ZIP) | Must enter value.<br><br>A subset of ADDRESS (ZIP) domain<br><br>On reference delete - Set null<br><br>On reference update - Cascade | Warehouse address zip code. This support the 5 character zip code format for US address |

## STORE_IN

Record the inventory stored in each warehouse

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **W_Code** | A fixed string of 5 characters | PK<br>FK references to WAREHOUSE (W_Code) | Must enter value.<br><br>A subset of WAREHOUSE (W_Code) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | Unique identifier for different warehouses |
| **Quantity** | Integer | | | Number of book stored in warehouse |

## RESTOCK_ID

Record the ordering transactions to publishers.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **R_ID** | Variable string of up to 10 characters | PK | Must enter value | Unique order ID to identify each transaction |
| **Date** | Date in the form of YYYY-MM-DD | | Must enter value | Date of order transaction submission |
| **Admin** | Variable string of up to 30 characters | | | Name of the staff who put in the order and entered the transaction |
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **Quantity** | Integer | | Must enter value | Number of book ordered |

## AUTHOR

Contains customer information, with email as unique identification for individual customer.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **Author_ID** | Variable string of up to 15 characters | PK | Must enter value | Unique identification assigned to author |
| **First_Name** | Variable string of up to 30 characters | | Must enter value | Author's first name |
| **MI** | Variable string of up to 30 characters | | | Author's middle name if known |
| **C_Phone** | Variable string of up to 30 characters | | Must enter value | Author's last name |

## WRITE

List the relationship between books and authors. Record the authors for each book. Each row represents unique combination of a book and one of its authors, if it has multiple authors.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **Author_ID** | Variable string of up to 15 characters | PK<br>FK references to AUTHOR (Author_ID) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | Unique identification assigned to author |

## RATE

Customers can optionally provide ratings for books. This relation stores how customers rate books.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **C_Email** | Variable string of up to 50 characters | FK references to CUSTOMER (C_Email) | A subset of CUSTOMER_INFO (C_Email) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | Customer who rated the book |

## CONSTITUTE

Relation that specifies the books in each customer's purchase order.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **ISBN** | Variable string of up to 13 characters | PK<br>FK references to BOOK (ISBN) | Must enter value<br><br>A subset of BOOK (ISBN) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | ISBN is unique book number assigned to individual books The data type supports both the 10 and 13 number formats |
| **Order_ID** | Variable string of up to 10 characters | PK<br>FK references to ORDERS | Must enter value<br><br>A subset of ORDERS (Order_ID) domain<br><br>On reference delete – Set null<br><br>On reference update - Cascade | Unique Order_ID to identify each transaction |
| **O_Quantity** | Integer | | Must enter value | Number of book ordered |

## C_LOCATE

Contains customer address information.

| Attribute Name | Data Type | Keys | Constraint | Description |
|---|---|---|---|---|
| **C_Email** | Variable string of up to 50 characters | PK<br>FK references to CUSTOMER_INFO (C_Email) | Must enter value<br><br>A subset of CUSTOMER_INFO (C_Email) domain<br><br>On reference delete - Cascade<br><br>On reference update - Cascade | Customer registered email for unique user account |
| **Address** | Variable string of up to 50 characters | FK references to ADDRESS (Address) for current US addresses | Must enter value.<br><br>A subset of ADDRESS (Address) domain<br><br>On reference delete - Set Null<br><br>On reference update - Cascade | Customer address, include street number and street name |
| **ZIP** | A fixed string of 5 characters | FK references to ADDRESS (ZIP) | Must enter value.<br><br>A subset of ADDRESS (ZIP) domain<br><br>On reference delete - Set null<br><br>On reference update - Cascade | Customer address zip code. This support the 5 character zip code format for US address |

# 2.0 SAMPLE SQL QUERIES

## 2.0 SAMPLE SQL QUERIES

### Sample Query 1: Find the titles of all books by Prachett that cost less than $10

**Expected outcome of query:**

A list of book titles written by Pratchett with sale price less than $10.

**Relational algebra:**

$$P\_AUTHOR \leftarrow \delta_{Last\_Name = Prachett}(AUTHOR)$$
$$P\_BOOK \leftarrow (WRITE \bowtie_{WRITE.Author\_ID = P\_AUTHOR.Author\_ID} P\_AUTHOR)$$
$$R1 \leftarrow (P\_BOOK \bowtie_{P\_BOOK.ISBN = BOOK.ISBN} BOOK)$$
$$RESULT \leftarrow \pi_{<Title>}(\delta_{Price < 10}(R1))$$

**SQL query:**

```
SELECT Title

FROM BOOK AS B, WRITE AS W, AUTHOR AS A

WHERE

    A.Last_Name = 'Prachett'        AND

    B.Price < 10                    AND

    B.ISBN = W.ISBN                 AND

   W.Author_ID = A.Author_ID;
```

## Sample Query 2: Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

**Expected outcome of query:**

Here we will use the customer 'Melanie' with the email "melanie@Duis.co.uk" as an example. The query will return a list of books that Melanie purchased along with their dates of purchase.

**Relational algebra:**

$S\_CUSTOMER \leftarrow (\delta_{C\_Email=melanie@Duis.co.uk}(CUSTOMER\_INFO))$

$R1 \leftarrow (S\_CUSTOMER \bowtie_{S\_CUSTOMER.C\_Email = ORDERS.C\_Email} ORDERS)$

$R2 \leftarrow (R1 \bowtie_{R1.Order\_ID = CONSTITUTE.Order\_ID} CONSTITUTE)$

$RESULT \leftarrow \pi_{<Title,Date>}(R2 \bowtie_{R2.ISBN = BOOK.ISBN} BOOK)$

**SQL query**

```
SELECT Title, Date

FROM CUSTOMER_INFO AS CU, BOOK AS B, ORDERS AS O, CONSTITUTE AS
C

WHERE

    CU.C_Email = 'melanie@Duis.co.uk'    AND

    CU.C_Email = O.C_Email               AND

    O.Order_ID = C.Order_ID              AND

    C.ISBN = B.ISBN;
```

## Sample Query 3: Find the titles and ISBNs for all books with less than 5 copies in stock

**Expected outcome of query:**

A list of books (title and ISBN) that has combined inventory of less than 5 copies.

**Relational algebra:**

$$\text{BOOK\_QUAN} \leftarrow \rho_{\text{R(ISBN, TotalQuan)}}(_{\text{ISBN}}\mathcal{F}_{\text{SUM Quantity}}(\text{STORE\_IN}))$$

$$\text{BOOK\_LESSTHAN5} \leftarrow \delta_{\text{TotalQuan<5}}(\text{BOOK\_QUAN})$$

$$\text{RESULT} \leftarrow \pi_{\text{ISBN, Title}}(\text{BOOK\_LESSTHAN5} \bowtie_{\text{BOOK\_LESSTHAN5.ISBN = BOOK.ISBN}} \text{BOOK})$$

**SQL query:**

```
SELECT B.Title, B.ISBN
FROM BOOK AS B, STORE_IN AS S
WHERE
            B.ISBN = S.ISBN
GROUP BY    S.ISBN
HAVING      SUM(S.Quantity) < 5;
```

## Sample Query 4: Give all the customers who purchased a book by King and the titles of King books they purchased

**Expected outcome of query:**

The names of customers that have bought a book by King and the titles of those books by King that they purchased.

**Relational algebra:**

$$P\_BOOKS \leftarrow \delta_{Last\ Name = King}(WRITE \bowtie_{WRITE.Author\_ID = AUTHOR.Author\_ID} AUTHOR)$$

$$T1 \leftarrow (P\_BOOKS \bowtie_{R\_BOOKS.ISBN = BOOK.ISBN} BOOK)$$

$$T2 \leftarrow (T1 \bowtie_{T1.ISBN = CONSTITUTE.ISBN} CONSTITUTE)$$

$$T3 \leftarrow (T2 \bowtie_{T2.Order\_ID = ORDERS.Order\_ID} ORDERS)$$

$$RESULT \leftarrow \pi_{C\_FName,\ C\_LName,\ Title}(CUSTOMER\_INFO \bowtie_{CUSTOMER\_INFO.Email = T3.Email} T3)$$

**SQL query:**

```
SELECT CI.C_FName, C_LName, B.Title

FROM BOOK AS B, WRITE AS W, AUTHOR AS A, ORDERS AS O, CONSTITUTE
AS C, CUSTOMER_INFO AS CI

WHERE

    O.Order_ID = C.Order_ID          AND

    C.ISBN = B.ISBN                  AND

    W.ISBN = B.ISBN                  AND

    A.Author_ID = W.Author_ID        AND

    A.Last_Name = 'King';
```

## Sample Query 5: Find the total number of books purchased by a single customer

**Expected outcome of query:**

This query will return with the number of books purchased by any individual customers as identified by their unique email. Here we will again use the customer 'Melanie' with the email "melanie@Duis.co.uk" as an example.

**Relational algebra:**

$$Q5R1 \leftarrow \delta_{C\_Email= \text{'melanie@Duis.co.uk'}} (ORDERS)$$
$$RESULT \leftarrow \mathcal{F}_{SUM\ Quantity} (Q5R1 \bowtie_{Q5R1.Order\_ID = CONSTITUTE.Order\_ID} CONSTITUTE)$$

**SQL query:**

```
SELECT    SUM(C.O_Quantity)
FROM      ORDERS AS O, CONSTITUTE AS C
WHERE     O.C_Email = 'melanie@Duis.co.uk' AND
          O.Order_ID = C.Order_ID;
```

## Sample Query 6: Find the customer who has purchased the most books and the total number of books they have purchased

**Expected outcome of query:**

This query will find the customer who has purchased the most books and the total number of books he/she has purchased. If there is more than one customer who bought the same number of books that meet this query, their names and email addresses along with total number of books bought will all be listed.

**Relational algebra:**

$$Q6R1 \leftarrow {}_{C\_Email}\mathcal{F}_{SUM\ O\_Quantity}\ (ORDERS \bowtie_{ORDERS.Order\_ID\ =\ CONSTITUTE.Order\_ID} CONSTITUTE)$$

$$Q6R2 \leftarrow \mathcal{F}_{MAX\ Quantity}\ (Q6R1)$$

$$RESULT\ (C\_Email, First\ Name, Last\ Name, Book\ Quantity) \leftarrow \pi_{C\_Email,\ C\_FName,\ C\_LName,\ SUM\ O\_Quantity}\ (CUSTOMER \bowtie_{CUSTOMER.C\_Email\ =\ Q6R2.C\_Email} Q6R2)$$

**SQL query:**

```
CREATE VIEW CUSTOMER_PURCHASE

AS SELECT C_Email, SUM(O_Quantity) AS BookTotal

    FROM ORDERS AS O, CONSTITUTE AS C

    WHERE       O.Order_ID = C.Order_ID

   GROUP BY     C_Email;


CREATE VIEW MAX_CUSTOMER

AS SELECT CP.C_Email, C_FName, C_LName, MAX (CP.BookTotal) as MaxBook

 FROM CUSTOMER_INFO AS CI, CUSTOMER_PURCHASE AS CP

    WHERE       CI.C_Email = CP.C_Email;


 CREATE VIEW MAX_CUSTOMERS

AS SELECT CP.C_Email, CI.C_FName, CI.C_LName, CP.BookTotal
```

```
FROM CUSTOMER_INFO AS CI, CUSTOMER_PURCHASE AS CP,
MAX_CUSTOMER as MC

        WHERE       CI.C_Email = CP.C_Email AND

                    CP.BookTotal = MC.MaxBook;
```

## Sample Query 7: List all book titles in the warehouse designated W_100

**Expected outcome of query:**

This query will return the titles of books that are currently stored in warehouse W_100, and the quantity of each book. The same query can be used for any of the warehouses by substituting the warehouse code.

**Relational algebra:**

$$\pi_{(\text{Title, Quantity})}( \text{BOOK} \bowtie_{(\text{BOOK.ISBN=STORE\_IN.ISBN})} ((\delta_{(\text{W\_Code='W\_100'})}(\text{STORE\_IN}))))$$

**SQL query:**

```
SELECT Title, Quantity
FROM BOOK AS B, STORE_IN as S
WHERE
    S.W_Code = 'W_100'          AND
    S.Quantity <> 0             AND
    B.ISBN = S.ISBN
```

## Sample Query 8: List all books on biography

**Expected outcome of query:**

This query will return the titles of all books that belong to the biography category. The same query can be used for any of the categories by substituting the category.

**Relational algebra:**

$$\pi_{(Title)}( BOOK \bowtie_{(BOOK.ISBN=CATEGORY.ISBN)} ((\delta_{(Category = \text{'Biography'})} CATEGORY)))$$

**SQL query:**

```
SELECT B.Title, B.ISBN

FROM BOOK AS B, CATEGORY as C, CATEGORY_KEY as CK

WHERE

    CK.Category = 'Biography'          AND

    CK.Category_ID = C.Category_ID     AND

    B.ISBN = C.ISBN;
```

## Sample Query 9: Compute the total number of books ordered from each publisher

**Expected outcome of query:**

This query will return the total number of books that were ordered from each publisher, listing in descending order. This is useful in negotiating future purchases.

**Relational algebra:**

$$Q9R1 \leftarrow {}_{ISBN}\mathcal{F}_{SUM\ Quantity}\ (RESTOCK\_ID)$$
$$RESULT \leftarrow {}_{P\_Name}\mathcal{F}_{SUM\ Quantity}\ (Q9R1)$$

**SQL query:**

```
CREATE VIEW PUBLISHER_ORDER

AS SELECT ISBN, SUM(Quantity) as BookQuantity

FROM RESTOCK_ID

GROUP BY ISBN;


SELECT BP.P_Name, SUM(PO.BookQuantity) as BookOrdered

FROM PUBLISHER_ORDER AS PO, BOOK_PUBLISHER AS BP

WHERE

        PO.ISBN = BP.ISBN

GROUP BY BP.P_Name

ORDER BY BookOrdered DESC;
```

**For additional sample queries, please refer to the file "SQL_QUERIES" contained in a separate text file.**

# 3.0 UPDATE OPERATIONS

# 3.0 UPDATE OPERATIONS

## 3.1 Inserting new data

Examples of how to insert new data are provided for books, publishers, authors and customers. The syntax for inserting new data to the database can be applied for the other tables by substituting the correct table and column names. A new entry is given as example in each of the cases.

### 3.1.1 Adding new books

```
INSERT INTO BOOK (ISBN, Title, Year, Format, Language, Price)
VALUES (‘1562344678’, 'Database for dummies',  '1990', 'Paperback', 'English',
'15.99');
```

ISBN has attributes that are foreign key that many other relations reference, therefore it must be entered prior to adding others including BOOK_PUBLISHER, RESTOCK_ID, RATE, CONSTITUTE, WRITE, CATEGORY, STORE_IN.

After new entry into BOOK, one needs to also make the following updates:

- Update AUTHOR (if new) before updating WRITE.
- Update PUBLISHER (if new) before updating BOOK_PUBLISHER.
- Update CATEGORY, STORE_IN, RESTOCK_ID

### 3.1.2 Adding new publishers

```
INSERT INTO PUBLISHER (P_Name, P_Email, P_Address, P_ZIP)
VALUES (‘NP Books', 'np@books.com', '13 S Neil Ave', 21283’);
```

PUBLISHER has attributes that are foreign keys to BOOK_PUBLISHER, PUBLISHER_PHONE and ADDRESS, therefore it must be entered prior to these relations.

After new entry into PUBLISHER, one needs to also make the following updates:

- Update PUBLISHER before updating BOOK_PUBLISHER.
- Update PUBLISHER_PHONE
- Update ADDRESS (if the address is new)

### 3.1.3 Adding new authors

```
INSERT INTO AUTHOR (Author_ID, First_Name, MI, Last_Name)
VALUES ('A0000021', ‘Brian’, NULL, 'Nelson');
```

After new entry into AUTHOR, one can then update the WRITE relation.

### 3.1.4 Adding new customers

```
INSERT INTO CUSTOMER_INFO (C_Email, C_Fname, C_Lname, C_Phone)
VALUES ('melanie@gmail.com', 'Melanie', English, '110-391-3995');
```

CUSTOMER_INFO has attributes as foreign keys to C_LOCATE, CUSTOMER_PAYMENT, ORDERS and RATE, therefore it must be entered prior to these relations.

After new entry into PUBLISHER, one needs to also make the following updates:

- Update CUSTOMER_PAYMENT (if credit card information provided)
- Update C_LOCATE (if known)
- Update ADDRESS (if the address is new)
- Update ORDERS (if customer makes new orders)

## 3.2 Deleting data

Examples of how to delete data are provided for books, publishers, authors and customers. The syntax for deleting specific rows of data from the database is as follows:

```
DELETE FROM <TABLE>
WHERE <Condition of rows to be deleted>;
```

However, care must be taken to ensure the integrity of the database, particularly as the deletion could impact other tables if it contains data that other table references. In general, deletion of data is not recommended.

### 3.2.1 Deleting books

```
DELETE FROM BOOK
WHERE ISBN = '0743455967';
```

This will delete the record of the book with ISBN = 0743455967. One can also choose to delete all books with certain features, e.g. all paperback (WHERE Format = 'Paperback') or any combination of features (Format = 'Paperback' AND Year = 2002).

For other tables that reference BOOK, constraint has been written into the script and therefore on deletion, those that refer to the ISBN of the BOOK will be set null.

### 3.2.2 Deleting publishers

```
DELETE FROM PUBLISHER
WHERE P_Name = 'Pocket Books';
```

For other tables that reference PUBLISHER (PUBLISHER_PHONE, BOOK_PUBLISHER), constraint has been written into the script and therefore on deletion, those that refer to the P_Name of the PUBLISHER will be set null.

### 3.2.3 Deleting authors

```
DELETE FROM AUTHOR
WHERE Author_ID = 'A0000001';
```

For the table that reference AUTHOR (WRITE), constraint has been written into the script and therefore on deletion, those that refer to the Author_ID of the AUTHOR will be set null.

### 3.2.4 Deleting customers

```
DELETE CUSTOMER_INFO
WHERE C_Email = 'melanie@Duis.co.uk';
```

For the tables that reference CUSTOMER_PAYMENT, C_LOCATE, RATE, ORDERS, constraints have been written into the script and therefore on deletion, those that refer to the C_Email of the CUSTOMER_INFO will be set null.