**CSE 3241 Final Project – The Final Report**

**Section 2. User Manual**

1. Table description

BOOK

* This table is for books to be sold/stored/refunded
* Isbn: VARCHAR(13), Primary key, NOT NULL
  + unique 10 or 13 digits to identify books
* Price: DECIMAL(7,2), NOT NULL
  + price of books
* Title: VARCHAR(40), NOT NULL
  + title of books
* Year: INT, NOT NULL
  + year of publication
* P\_name: VARCHAR(20), Foreign key, NOT NULL
  + publisher name of books

GENRE

* This table contains the genre(s) of books in the bookstore
* B\_Isbn: VARCHAR(13), Primary key, Foreign key, NOT NULL
  + unique 10 to 13 characters to identify books; references Isbn in book table
* Genre: VARCHAR(20), primary key, NOT NULL
  + name of book genre

AUTHOR

* This table contains authors of a book or books in the bookstore
* Id: VARCHAR(5), Primary key, NOT NULL
  + unique identifier for each author, which cannot exceed 5 characters
* Fname: VARCHAR(20), NOT NULL
  + The author’s first name
* Lname: VARCHAR(20), NOT NULL
  + The author’s last name
* Mname: VARCHAR(20)
  + The author's middle name / initial
* Bdate: DATE
  + THe author’s date of birth

WAREHOUSE

* This table contains warehouses that store book(s) in the database
* Id: VARCHAR(5), Primary key, NOT NULL
  + unique identifier for each warehouse, which cannot exceed 5 characters
* Location: VARCHAR(20), NOT NULL
  + The name of the city that the warehouse is in
* Capacity: INT, NOT NULL
  + The number of books that can be stored at the warehouse

CUSTOMER

* This table contains customers that have placed an order with the bookstore
* Username: VARCHAR(20), Primary key, NOT NULL
  + A unique username, which cannot exceed 20 characters, that each customer creates to place orders with the online bookstore
* Password: VARCHAR(10), NOT NULL
  + The customer’s password to log into their bookstore account
* Email: VARCHAR(40), NOT NULL
  + The customer’s email address, where they receive order and refund details and updates
* Phone\_num: INT, NOT NULL
  + The customer’s phone number

PUBLISHER

* This table contains publishing company of a book or books in the bookstore
* Name: VARCHAR(20), Primary key, NOT NULL
  + The official name of the publishing company
* Email: VARCHAR(40), NOT NULL
  + The email address where the publishing company can be contacted

ORDERS

* This table contains orders placed by customers with the bookstore
* Order\_num: CHAR(10), Primary key, NOT NULL
  + A unique 10 character identifier for each order
* Order\_date: DATE, NOT NULL
  + The date that the order was placed
* Ship\_address: VARCHAR(100), NOT NULL
  + The full shipping address for the order, including street address, city, state, and zip code
* Customer\_username: VARCHAR(20), Foreign key, NOT NULL
  + A unique username, which cannot exceed 20 characters, that each customer creates to place orders with the online bookstore; references Username in CUSTOMER table
* Creditcard\_number: INT, Foreign key
  + The number of the credit card used to place the order; only present if the customer used a credit card to place the order; references Number in CREDIT\_CARD table
* Paypal\_username: VARCHAR(20), Foreign key
  + The username of the PayPal account used to place the order; only present if the customer used a PayPal account to place the order; references Username in PAYPAL table

CREDIT\_CARD

* This table contains credit cards used to place an order or orders with the bookstore
* Number: INT, Primary key, NOT NULL
  + The credit card’s number, which is unique for each credit card
* Company: VARCHAR(20), NOT NULL
  + The company that issued the credit card
* Cvv: INT, NOT NULL
  + The three or four digit CVV, or card verification value, on the credit card
* Name: VARCHAR(20), NOT NULL
  + The cardholder name, or the person who owns the credit card
* Bill\_address: VARCHAR(40), NOT NULL
  + The billing address associated with the credit card

PAYPAL

* This table contains PayPal accounts used to place an order or orders with the bookstore
* Username: VARCHAR(20), Primary key, NOT NULL
  + A unique username, which cannot exceed 20 characters, that each PayPal user creates to make payments with PayPal
* Password: VARCHAR(10), NOT NULL
  + The PayPal user’s password to log into their PayPal account

STORES

* This table contains the quantity of a specific book stored at a specific warehouse
* B\_isbn: VARCHAR(13), Primary key, Foreign key, NOT NULL
  + unique 10 to 13 character identifier for each book; references Isbn in the BOOK table
* W\_id: VARCHAR(5), Primary key, Foreign key, NOT NULL
  + unique identifier for each warehouse, which cannot exceed 5 characters; references Id in the WAREHOUSE table
* Quantity: INT, NOT NULL
  + The number of copies of the specific book stored at the warehouse

WRITES

* This table contains the relationships between books and authors, in other words: which authors have written which books
* B\_isbn: VARCHAR(13), Primary key, Foreign key, NOT NULL
  + unique 10 to 13 character identifier for each book; references Isbn in the BOOK table
* A\_id: VARCHAR(5), Primary key, Foreign key, NOT NULL
  + unique identifier for each author, which cannot exceed 5 characters; references Id in the AUTHOR table

ADDED\_TO

* This table contains which books have been added to certain orders and the quantity of copies added
* B\_isbn: VARCHAR(13), Primary key, Foreign key, NOT NULL
  + unique 10 to 13 character identifier for each book; references Isbn in the BOOK table
* O\_order\_num: CHAR(10), Primary key, Foreign key, NOT NULL
  + A unique 10 character identifier for each order; references Order\_num in the ORDERS table
* Quantity: INT, NOT NULL
  + The number of copies of the specific book added to the order

REFUND

* This table contains the refunds of orders placed with the bookstore
* Order\_num: CHAR(10), Primary key, Foreign key, NOT NULL
  + A unique 10 character identifier for each order; references Order\_num in the ORDERS table
* Item: VARCHAR(20), Primary key, NOT NULL
  + The name (title or Isbn) of the item (book) to be refunded in the order
* Count: INT, NOT NULL
  + Number of item (books) to be refunded in the order
* Reason: TEXT
  + customer’s stated reason for the refund, which can be left empty
* Method: CHAR(2), NOT NULL
  + The method of delivering the refund; ‘PO’ means point and ‘MO’ means money (credit card)

1. Sample SQL queries

* Find the titles of all books by Pratchett that cost less than $10

SELECT Title

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

WHERE B.Isbn=W.B\_isbn AND A.Id=W.A\_Id

AND A.Lname = 'Pratchett'

AND B.Price < 10.0;

* Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customers – Gilbert\_0)

SELECT B.Title, C.Username, O.Order\_date

FROM BOOK AS B, CUSTOMER AS C, ADDED\_TO AS A, ORDERS AS O

WHERE B.Isbn=A.B\_isbn

AND A.O\_order\_num = O.Order\_num

AND C.Username = O.Customer\_username

AND C.Username = 'Gilbert\_0';

* 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 B.Isbn = S.B\_isbn

AND S.Quantity < 5;

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

SELECT C.Username, B.Title

FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O, BOOK AS B, AUTHOR AS A, WRITES AS W

WHERE C.Username=O.Customer\_username

AND O.Order\_num=AT.O\_order\_num

AND B.Isbn=AT.B\_isbn

AND B.Isbn=W.B\_isbn

AND W.A\_id=A.Id

AND A.Lname = 'Pratchett';

* Find the total number of books purchased by a single customer (you choose how to designate the customer – Gilbert\_0)

SELECT SUM(AT.Quantity)

FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username

AND O.Order\_num=AT.O\_order\_num

AND C.Username = 'Gilbert\_0'

GROUP BY C.Username;

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

SELECT Username, SUM(AT.Quantity)

FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username

AND O.Order\_num=AT.O\_order\_num

GROUP BY C.Username

HAVING SUM(AT.Quantity) IN (

SELECT MAX(Total\_book)

FROM (SELECT SUM(AT.Quantity) AS Total\_book

FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username

AND O.Order\_num=AT.O\_order\_num

GROUP BY C.Username)

);

* Retrieve the usernames of customers who use Paypal for payment

SELECT DISTINCT(C.Username)

FROM CUSTOMER AS C, ORDERS AS O

WHERE C.Username = O.Customer\_username

AND O.Creditcard\_number IS NULL;

* Find the warehouse with the greatest quantity of a certain book (ISBN = 1579902944)

SELECT W.Id, W.Location

FROM STORES AS S, WAREHOUSE AS W

WHERE W.Id = S.W\_id AND S.B\_isbn = '1579902944'

GROUP BY S.B\_isbn

HAVING S.Quantity = MAX(S.Quantity);

* Find all the items to refunded on orders from a specific date (Order\_date = '2022-10-10')

SELECT DISTINCT R.Item AS refunded\_book\_isbn

FROM REFUND AS R, ORDERS AS O

WHERE R.Order\_num = O.Order\_num AND O.Order\_date = '2022-10-10';

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

SELECT C.Username, SUM(AT.Quantity\*B.Price) AS Total\_Dollar

FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username

AND O.Order\_num = AT.O\_order\_num

AND AT.B\_isbn = B.Isbn

GROUP BY C.Username;

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

SELECT T.Name, T.Email, T.Each\_total

FROM (SELECT C.Username AS Name, C.Email AS Email, SUM(AT.Quantity\*B.Price) AS Each\_total

FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username

AND O.Order\_num = AT.O\_order\_num

AND AT.B\_isbn = B.Isbn

GROUP BY C.Username) AS T

GROUP BY T.Name

HAVING T.Each\_total > AVG((SELECT SUM(AT.Quantity\*B.Price)

FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username

AND O.Order\_num = AT.O\_order\_num

AND AT.B\_isbn = B.Isbn

GROUP BY C.Username));

* 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 B.Title, SUM(AT.Quantity) AS total\_copies

FROM BOOK AS B, ADDED\_TO AS AT

WHERE B.Isbn = AT.B\_isbn

GROUP BY B.Title

ORDER BY total\_copies DESC;

* 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 B.Title, SUM(AT.Quantity)\*B.Price AS dollar\_totals

FROM BOOK AS B, ADDED\_TO AS AT

WHERE B.Isbn = AT.B\_isbn

GROUP BY B.Title

ORDER BY dollar\_totals DESC;

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

SELECT A.FName, A.LName

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

WHERE B.Isbn=W.B\_isbn

AND A.Id=W.A\_Id

AND B.Isbn IN (SELECT AT.B\_isbn

FROM ADDED\_TO AS AT

GROUP BY AT.B\_isbn

HAVING SUM(AT.Quantity) >= (SELECT MAX(Summation.S)

FROM (SELECT SUM(Quantity) AS S FROM ADDED\_TO GROUP BY B\_isbn) AS Summation)

);

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

SELECT A.FName, A.LName

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

WHERE B.Isbn=W.B\_isbn

AND A.Id=W.A\_Id

AND B.Isbn IN (SELECT AT.B\_isbn

FROM ADDED\_TO AS AT, BOOK AS B

WHERE B.Isbn=AT.B\_isbn

GROUP BY AT.B\_isbn

HAVING SUM(AT.Quantity)\*B.Price >= (SELECT MAX(Summation.D)

FROM (SELECT SUM(AT.Quantity)\*B.Price AS D FROM ADDED\_TO AS AT, BOOK AS B WHERE B.Isbn=AT.B\_isbn GROUP BY AT.B\_isbn) AS Summation)

);

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

SELECT DISTINCT(O.Customer\_username) AS username

FROM ORDERS AS O, BOOK AS B, WRITES AS W, ADDED\_TO AS AT

WHERE O.Order\_num=AT.O\_order\_num AND B.Isbn=AT.B\_isbn AND W.B\_isbn=B.Isbn AND

W.A\_id IN (

SELECT A.Id

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

WHERE B.Isbn=W.B\_isbn

AND A.Id=W.A\_Id

AND B.Isbn IN (SELECT AT.B\_isbn

FROM ADDED\_TO AS AT, BOOK AS B

WHERE B.Isbn=AT.B\_isbn

GROUP BY AT.B\_isbn

HAVING SUM(AT.Quantity)\*B.Price >= (SELECT MAX(Summation.D)

FROM (SELECT SUM(AT.Quantity)\*B.Price AS D FROM ADDED\_TO AS AT, BOOK AS B WHERE B.Isbn=AT.B\_isbn GROUP BY AT.B\_isbn) AS Summation)

));

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

SELECT DISTINCT A.Fname, A.Lname

FROM ADDED\_TO AS AT, BOOK AS B, WRITES AS W, AUTHOR AS A, ORDERS AS O

WHERE AT.B\_isbn=B.Isbn AND B.Isbn=W.B\_isbn AND W.A\_id=A.Id AND O.Order\_num=AT.O\_order\_num AND

O.Customer\_username IN (

SELECT T.Name

FROM (SELECT C.Username AS Name, O.Order\_num AS Onum, SUM(AT.Quantity\*B.Price) AS Each\_total

FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username

AND O.Order\_num = AT.O\_order\_num

AND AT.B\_isbn = B.Isbn

GROUP BY C.Username) AS T

GROUP BY T.Name

HAVING T.Each\_total > AVG((SELECT SUM(AT.Quantity\*B.Price)

FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username

AND O.Order\_num = AT.O\_order\_num

AND AT.B\_isbn = B.Isbn

GROUP BY C.Username)));

1. Insert queries

Adding new books

INSERT INTO BOOK VALUES (‘Isbn’, Price, ‘Title’, ‘P\_name’);

INSERT INTO GENRE VALUES ('B\_isbn', 'Genre');

INSERT INTO STORES VALUES ('B\_isbn', 'W\_id1', Quantity), ('B\_isbn', 'W\_id2', Quantity) …;

Isbn must be unique

Isbn, Price, Title, and P\_name are not null

P\_name refers to PUBLISHER’s Name

* It is corresponded to one of Names in the PUBLISHER
* If the publisher’s name is not in the database, publisher name is added first in the PUBLISHER table

Isbn will be referred to GENRE, STORES, WRITES, and ADDED\_TO

* After inserting a new book, its writers need to be connected
* If the author’s name is not in the database, author’s information is added first in the AUTHOR table
* After inserting a new book, its genre can be inserted
* After inserting a new book, warehouses that store the book need to be noticed

Adding new publishers

INSERT INTO PUBLISHER VALUES (‘Name’, ‘Email’);

Name must be unique

Name, and Email are not null

Name will be referred to BOOK

Adding new authors

INSERT INTO AUTHOR VALUES (‘Id’, ‘Fname’, ‘Lname’, ‘Mname’, Bdate);

INSERT INTO WRITES VALUES (‘B\_isbn’, 'A\_id1'), (‘B\_isbn’, 'A\_id2'), (‘B\_isbn’, 'A\_id3') …;

Id must be unique

Id, Fname, and Lname are not null

Id will be referred to WRITES

* After inserting a new book & its author(s), they need to be connected through WRITES

Adding new customers

INSERT INTO CUSTOMER VALUES (‘Username’, ‘Password’, ‘Email’, ‘Phone\_num’);

Username must be unique

Username, Password, Email, and Phone\_num are not null

Username will be referred to ORDERS

1. Delete queries

Removing books

DELETE FROM WRITES WHERE B\_isbn = ‘xxxxxxxxxx’

DELETE FROM GENRE WHERE B\_isbn = ‘xxxxxxxxxx’

DELETE FROM STORES WHERE B\_isbn = ‘xxxxxxxxxx’

DELETE FROM ADDED\_TO WHERE B\_isbn = ‘xxxxxxxxxx’

DELETE FROM BOOK WHERE Isbn = ‘xxxxxxxxxx’ (number of digits = 10)

Or

DELETE FROM WRITES WHERE B\_isbn = ‘xxxxxxxxxxxxx’

DELETE FROM GENRE WHERE B\_isbn = ‘xxxxxxxxxxxxx’

DELETE FROM STOTES WHERE B\_isbn = ‘xxxxxxxxxxxxx’

DELETE FROM ADDED\_TO WHERE B\_isbn = ‘xxxxxxxxxxxxx’

DELETE FROM BOOK WHERE Isbn = ‘xxxxxxxxxxxxx’ (number of digits = 13)

Must first delete the rows with the matching Isbn from the tables referencing the corresponding Isbn of the book you want to delete

Removing publishers

DELETE FROM WRITES WHERE B\_isbn IN (SELECT Isbn FROM BOOK WHERE P\_name=’…’)

DELETE FROM GENRE WHERE B\_isbn IN (SELECT Isbn FROM BOOK WHERE P\_name=’…’)

DELETE FROM STORES WHERE B\_isbn IN (SELECT Isbn FROM BOOK WHERE P\_name=’…’)

DELETE FROM ADDED\_TO WHERE B\_isbn IN (SELECT Isbn FROM BOOK WHERE P\_name=’…’)

DELETE FROM BOOK WHERE P\_name=’…’

DELETE FROM PUBLISHER WHERE Name=’…’

Must first delete the rows with the matching Isbn from the tables referencing the corresponding Isbn of the book written by a publisher you want to delete

Must second delete rows with the matching (publisher) Name from the tables referencing the corresponding (publisher) Name of the publisher you want to delete

Removing authors

DELETE FROM WRITES WHERE A\_id=’…’

DELETE FROM AUTHOR WHERE Id=’…’

Must first delete the rows with the matching (author) Id from WRITES referencing the corresponding (author) Id of the author you want to delete

Removing customers

DELETE FROM ADDED\_TO WHERE O\_order\_num IN (SELECT Order\_num FROM ORDERS WHERE Customer\_username=’…’)

DELETE FROM REFUND WHERE O\_order\_num IN (SELECT Order\_num FROM ORDERS WHERE Customer\_username=’…’)

DELETE FROM ORDERS WHERE Customer\_username =’…’

DELETE FROM CUSTOMER WHERE Username=’…’

Must first delete the rows with the matching order number from the tables referencing the corresponding order numbers of the ORDERS ordered by a customer you want to delete

Must second delete rows with the matching customer username from the tables referencing the corresponding customer username of the CUSTOMER you want to delete