

Introduction to Database Systems - CSC 675/775

Spring 2022

PROJECT TITLE: ONLINE BOOK STORE APPLICATION

Team Members: Hirva Patel
Aarshil Patel
Jasmine Jahan
Umar Rama
Gabriel Gonzalez

Sr No.	Content	Page No
1	Create Table statements, constraints, indexes, views	2
2	Queries and Screenshots	6
3	Tables Screenshots: All the select queries	10

1. Create Tables, statements , constraints, indexes, views :

```
CREATE TABLE if not exists Books(  
    ISBN VARCHAR(45) NOT NULL AUTO_INCREMENT,  
    price FLOAT,  
    title VARCHAR(32) NOT NULL,  
    category VARCHAR(32),  
    author VARCHAR(32),  
    year YEAR(4)  
    PRIMARY KEY (ISBN)  
);
```

```
CREATE TABLE if not exists User (  
    user_id INT,  
    name VARCHAR(32) NOT NULL,  
    address VARCHAR(32),  
    email VARCHAR(32),  
    phone VARCHAR(32)  
    PRIMARY KEY (user_id)  
);
```

```
CREATE TABLE Cart (  
    cartId INT NOT NULL,  
    total_price INT NOT NULL,  
    quantity INT NOT NULL,  
    PRIMARY KEY (cartId)  
);
```

```
CREATE TABLE Has (  
    user_id INT NOT NULL,  
    cartId INT NOT NULL,  
    PRIMARY KEY (userId, cartId),  
    INDEX cartId_idx (cartId ASC) VISIBLE,  
    FOREIGN KEY (userId) REFERENCES User (userId) ON DELETE CASCADE  
    FOREIGN KEY (cartId) REFERENCES Cart (cartId) ON DELETE CASCADE  
)
```

```

CREATE TABLE Admin (
    username VARCHAR(45) NOT NULL,
    password VARCHAR(45) NOT NULL,
    PRIMARY KEY (username)
);

CREATE TABLE Update (
    username VARCHAR(45) NOT NULL,
    ISBN INT NOT NULL,
    PRIMARY KEY (username, ISBN),
    FOREIGN KEY (username) REFERENCES Admin (username) ON DELETE
    CASCADE
);

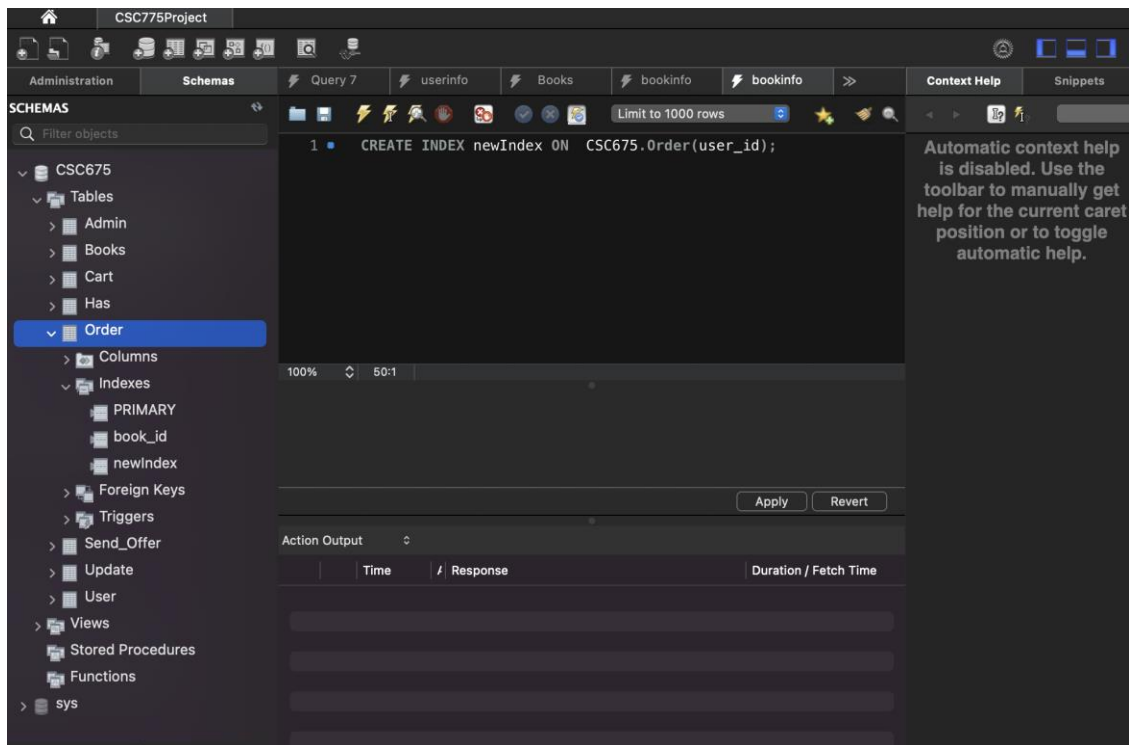
CREATE TABLE Send_Offer (
    offerid INT NOT NULL,
    user_id INT NOT NULL,
    username VARCHAR(45) NOT NULL,
    PRIMARY KEY (offerid, user_id, username),
    CONSTRAINT `customername`
    FOREIGN KEY (`username`) REFERENCES `Admin` (`username`) ON
    DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT `person_id`
    FOREIGN KEY (`user_id`) REFERENCES `User` (`user_id`) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE Order (
    Dateoforder date NOT NULL,
    user_id int NOT NULL,
    ISBN varchar(45) NOT NULL,
    Quantity int NOT NULL,
    PRIMARY KEY (`Dateoforder`, `user_id`, `ISBN`),
    CONSTRAINT `book_id`
    FOREIGN KEY (`ISBN`) REFERENCES `Books` (`ISBN`) ON DELETE
    CASCADE ON UPDATE CASCADE,
    CONSTRAINT `customer_id`
    FOREIGN KEY (`user_id`) REFERENCES `User` (`user_id`) ON DELETE
    CASCADE ON UPDATE CASCADE)

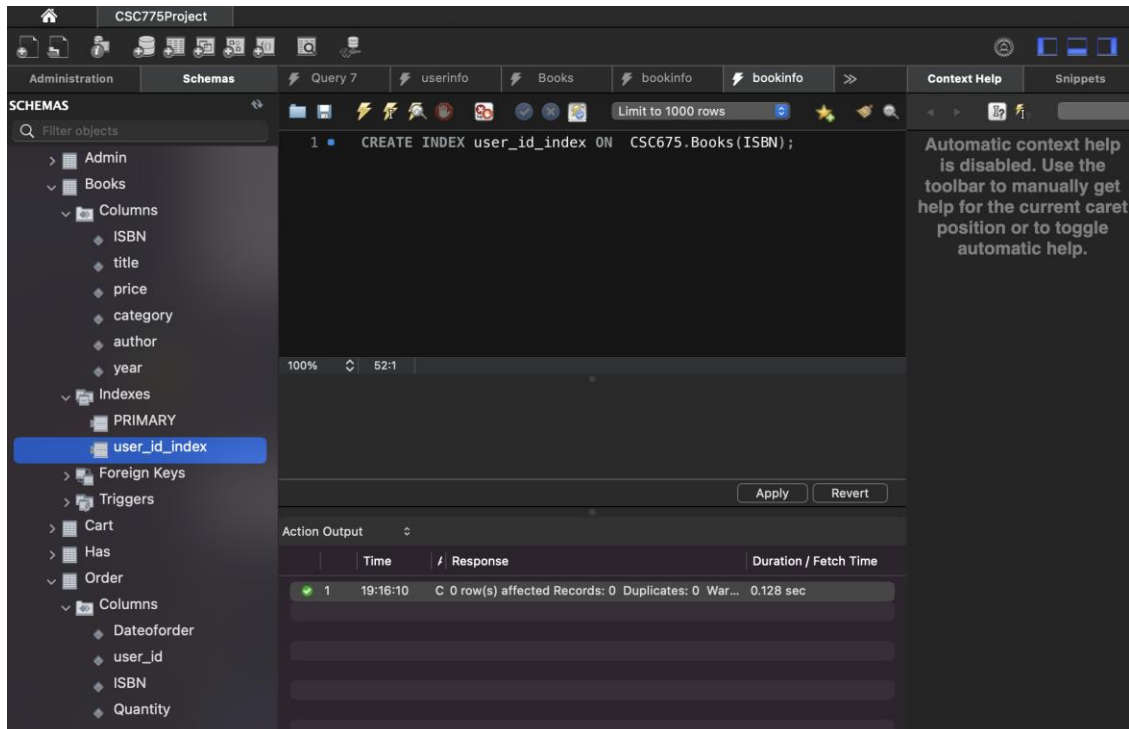
```

Index:

CREATE INDEX newIndex ON Order(user_id);



CREATE INDEX user_id_index ON CSC675.Books(ISBN);



Views:

<pre> CREATE VIEW `CSC675`.`Books_sold` AS SELECT `CSC675`.`Books`.`title` AS `title`, `CSC675`.`Order`.`ISBN` AS `ISBN`, SUM(`CSC675`.`Order`.`Quantity`) AS `books_sold` FROM (`CSC675`.`Order` JOIN `CSC675`.`Books`) WHERE (`CSC675`.`Order`.`ISBN` = `CSC675`.`Books`.`ISBN`) GROUP BY `CSC675`.`Order`.`ISBN` ORDER BY `books_sold` </pre>	<pre> CREATE VIEW CSC675.userinfo AS SELECT email FROM CSC675.User, CSC675.Order WHERE CSC675.User.user_id = CSC675.Order.user_id; </pre>
--	---

1 • **SELECT * FROM CSC675.Books_sold;**

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	title	ISBN	books_sold
▶	Dbms	222	2
	Algorithm	333	2
	Grit	111	5

Books_sold 1 x

Output

Action Output

#	Time	Action	Message
✓ 1	17:03:15	SELECT * FROM CSC675.Books_sold LIMIT 0, 1000	3 row(s) retu

2. Queries and Screenshots

1. Display the name of all the users grouped by user id.

```
SELECT * FROM User;  
SELECT name, address  
FROM User  
GROUP BY name  
HAVING COUNT(*) < 5;
```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The query editor contains the following SQL code:

```
1 SELECT * FROM CSC675.User;  
2 SELECT name, address  
3 FROM CSC675.User  
4 GROUP BY name  
5 HAVING count(*) < 5;
```

Below the query editor, the 'Result Grid' is displayed with columns 'name' and 'address'. It shows three rows of data:

	name	address
▶	Thomas	San Jose
	John	Santa Clara
	Mary	San Francisco

At the bottom, there are tabs for 'User 12' and 'User 13', and an 'Action Output' section.

2. Display the title and author of all the books where the price of the books is greater than 15 and the count is less than 5.

```
SELECT * FROM Books;
SELECT title, author
FROM Books
WHERE price > 15
GROUP BY ISBN
HAVING COUNT(*) < 5;
```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The query editor contains the following SQL code:

```
1 SELECT * FROM CSC675.Books;
2 SELECT title, author
3 FROM CSC675.Books
4 WHERE price > 15
5 GROUP BY ISBN
6 HAVING COUNT(*) < 5;
```

Below the editor, the 'Result Grid' tab is active, displaying a table with the following data:

	title	author
▶	Grit	Angela
▶	Dbms	Gehrke
▶	Algorithm	Nepolean

At the bottom, the 'Action Output' tab is visible but empty.

3. Display names of users who ordered books having ISBN of 333.

```

SELECT * FROM Order;
SELECT name, user_id
FROM User
WHERE user_id IN ( SELECT user_id
                    FROM Order
                    WHERE ISBN = 333);

```

The screenshot shows a database management tool interface. At the top, a SQL query is entered in a text area:

```

1 SELECT * FROM CSC675.Order;
2 SELECT name, user_id
3 FROM CSC675.User
4 WHERE user_id IN ( SELECT user_id
5 FROM CSC675.Order
6 WHERE ISBN = 333);
7

```

Below the query editor, the "Result Grid" is displayed, showing the results of the query. The grid has two columns: "name" and "user_id". The results are as follows:

name	user_id
John	13
Mary	14

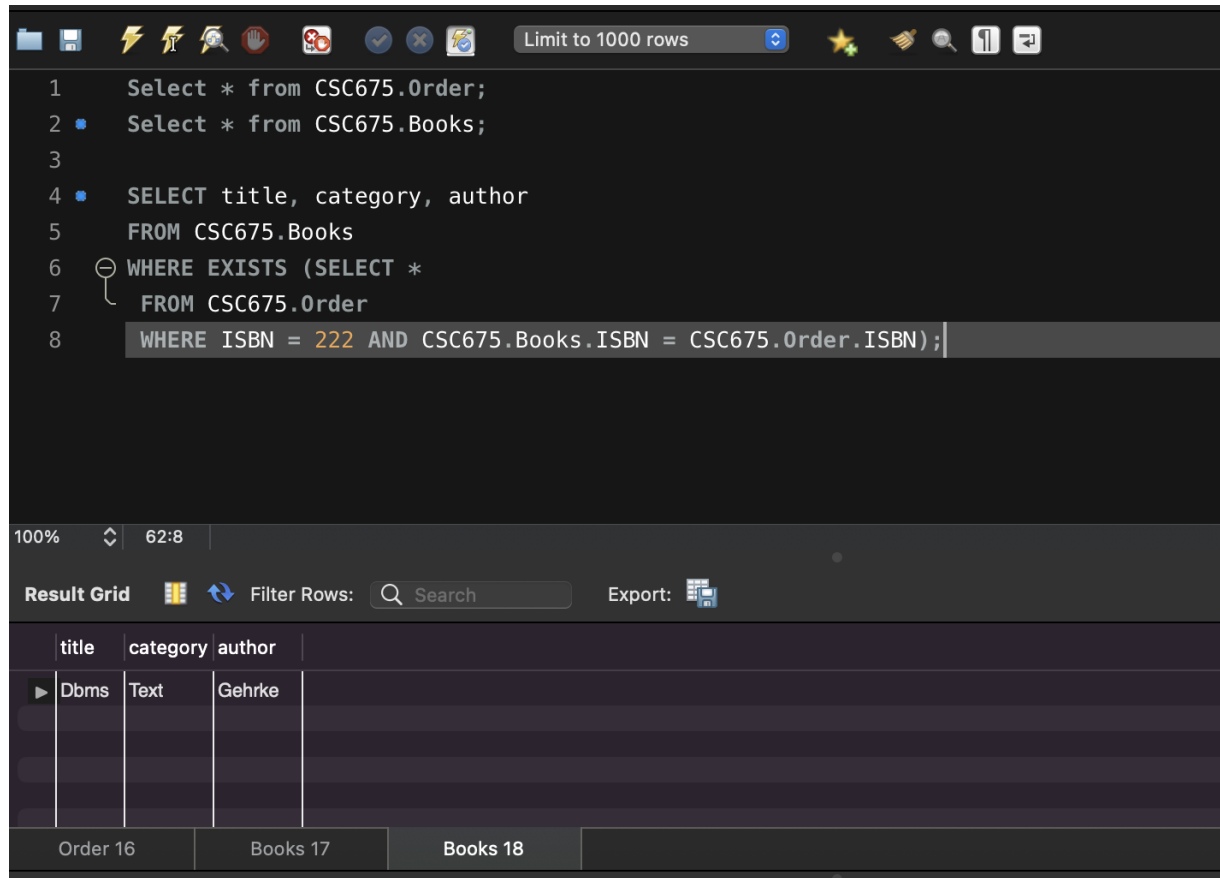
At the bottom of the interface, the "Output" pane shows the execution log. It includes a table with columns: "#", "Time", "Action", and "Message". The log shows the following entries:

#	Time	Action	Message
5	11:46:32	SELECT * FROM CSC675.Update LIMIT 0, 1000	3 row(s) returned
6	11:46:38	SELECT * FROM CSC675.Order LIMIT 0, 1000	7 row(s) returned
7	11:46:56	SELECT * FROM CSC675.Books LIMIT 0, 1000	3 row(s) returned
8	11:48:45	SELECT * FROM CSC675.Order LIMIT 0, 1000	7 row(s) returned
9	11:48:45	SELECT name FROM CSC675.User WHERE user_id IN (SELECT user_id FROM CSC675.Order ...	2 row(s) returned
10	11:49:16	SELECT * FROM CSC675.Order LIMIT 0, 1000	7 row(s) returned
11	11:49:16	SELECT name, user_id FROM CSC675.User WHERE user_id IN (SELECT user_id FROM CSC675...	2 row(s) returned

4. Display title, category and author of books that were ordered and have

ISBN as 222.

```
SELECT title, category, author
FROM Books
WHERE EXISTS ( SELECT *
FROM Order
WHERE ISBN = 222 AND Books.ISBN = Order.ISBN);
```



3. Tables Screenshots: All the select queries

Limit to 1000 rows

```
1 SELECT * FROM CSC675.User;
```

00% 27:1

Result Grid Filter Rows: Search Edit: Export/Import:

	user_id	name	address	email	phone
▶	12	Thomas	San Jose	tho@gmail.com	100-294-3539
	13	John	Santa Clara	john@yahoo.com	200-567-5555
	14	Mary	San Francisco	mary@hotmail.com	400-999-1234
	15	Mary	Palo Alto	m@gmail.com	555-678-9999
	HULL	HULL	HULL	HULL	HULL

User 6

Limit to 1000 rows

```
1 SELECT * FROM CSC675.Update;
```

00% 29:1

Result Grid Filter Rows: Search Edit: Export/Import:

	username	ISBN
▶	aria	111
	john	222
	mary	333
	HULL	HULL

Update 5

Limit to 1000 rows

```
1 • SELECT * FROM CSC675.Send_Offer;
```

00% 33:1

Result Grid Filter Rows: Search Edit: Export/Import:

	offerid	user_id	username
▶ 22	0		
22	12	Thomas	
33	13	John	
44	14	Mary	
	NULL	NULL	NULL

Send_Offer 4

Limit to 1000 rows

```
1 • SELECT * FROM CSC675.Order;
```

Result Grid Filter Rows: Search Edit: Export/Import: Wrap Cell Content:

	Dateoforder	user_id	ISBN	Quantity
▶	2002-05-04	12	111	2
	2002-08-02	13	111	1
	2002-08-02	13	222	1
	2002-08-02	13	333	1
	2020-04-03	14	111	2
	2020-04-03	14	222	1
	2020-04-03	14	333	1
*	NULL	NULL	NULL	NULL

Order 2 ×

Output

Action Output

#	Time	Action	Message
✓ 11	11:49:16	SELECT name, user_id FROM CSC675.User WHERE user_id IN (SELECT user_id FROM CSC675...	2 row(s) returned
✓ 12	11:53:56	SELECT * FROM CSC675.Order LIMIT 0, 1000	7 row(s) returned
✓ 13	11:53:56	SELECT * FROM CSC675.Books LIMIT 0, 1000	3 row(s) returned
✓ 14	11:53:56	SELECT title, category, author FROM CSC675.Books WHERE EXISTS (SELECT * FROM CSC675....	1 row(s) returned

