Name:

Class:

Roll No:

**ONLINE BOOKSTORE SYSTEM**

Contents

[Part A: Analysis and Design 2](#_Toc46338689)

[Part B: SQL Statemenṭs 10](#_Toc46338690)

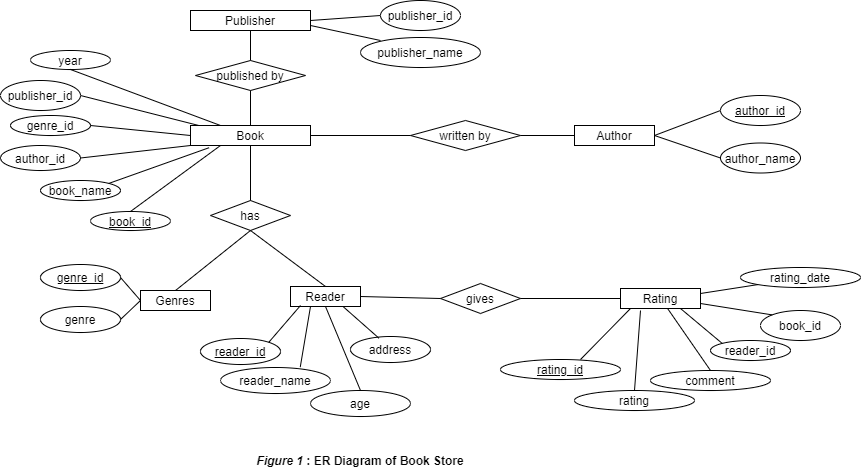
[Conclusion 13](#_Toc46338691)

## Part A: Analysis and Design

**Question : Design a database for maintaining book reviews. The database should contain details about each book: name (unique), author, genre, publisher and year. The database should also capture the details of each reader: name (unique), address and age. The reader can rate any book on a scale of 1-10 and provide comments (text). A reader can review a particular book more than once, i.e. on different dates.**

**Question 1:Draw an ER diagram for the system. Make sure to indicate primary keys, cardinality constraints, weak entities (if any) and any constraints. List any assumptions you make in the process. Note that you may need to define more attributes in order to be able to represent the given requirements.**

**Answer 1:**

****

**Question 2: Based on the ER diagram, design a well normalised (3NF) relational database to store the information. Make sure that the translation captures key constraints (primary keys, foreign keys, etc). You must provide the relational schema for this database and then implement it in a DBMS such as MySQL.**

**Answer 2: Creating normalised tables using MySQL.**

**1.AUTHOR**

CREATE TABLE AUTHOR(

AUTHOR\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\* Author ID \*/

AUTHOR\_NAME VARCHAR(50) NOT NULL, /\* Author Name \*/

PRIMARY KEY(AUTHOR\_ID)

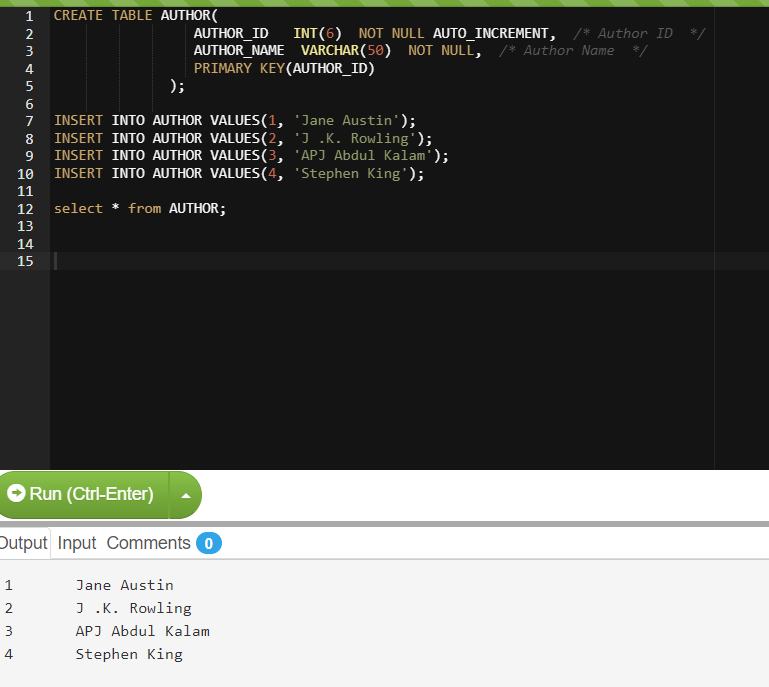
);

**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO AUTHOR VALUES(1, 'Jane Austin');

INSERT INTO AUTHOR VALUES(2, 'J .K. Rowling');

INSERT INTO AUTHOR VALUES(3, 'APJ Abdul Kalam');

INSERT INTO AUTHOR VALUES(4, 'Stephen King');

**2.GENRES**

CREATE TABLE GENRES(

GENRE\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\* Genre ID \*/

GENRES VARCHAR(50) NOT NULL, /\* Genre \*/

PRIMARY KEY(GENRE\_ID)

);

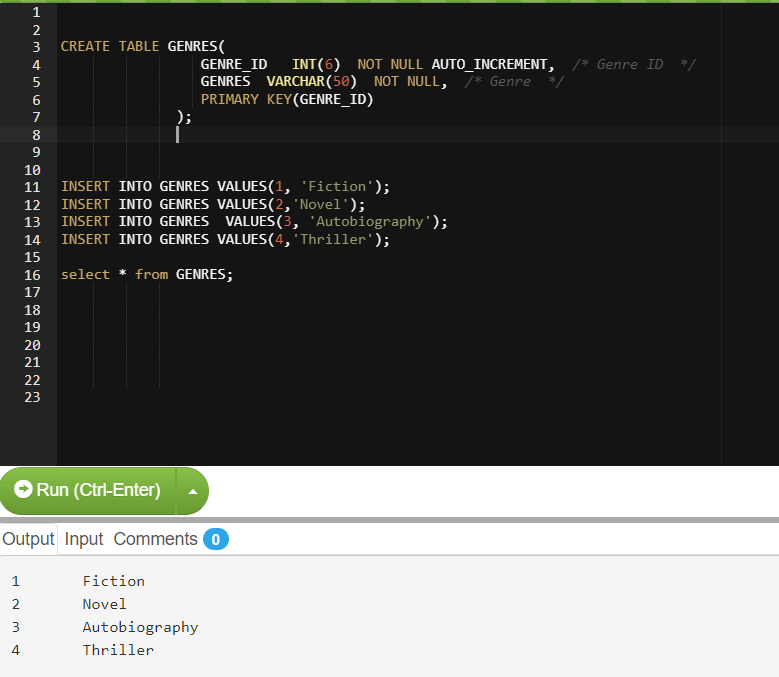
**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO GENRES VALUES(1, 'Fiction');

INSERT INTO GENRES VALUES(2,'Novel');

INSERT INTO GENRES VALUES(3, 'Autobiography');

INSERT INTO GENRES VALUES(4,'Thriller');



**3.PUBLISHER**

CREATE TABLE PUBLISHER(

PUBLISHER\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\* Publisher ID \*/

PUBLISHER\_NAME VARCHAR(50) NOT NULL, /\* Publisher Name \*/

PRIMARY KEY(PUBLISHER\_ID)

);

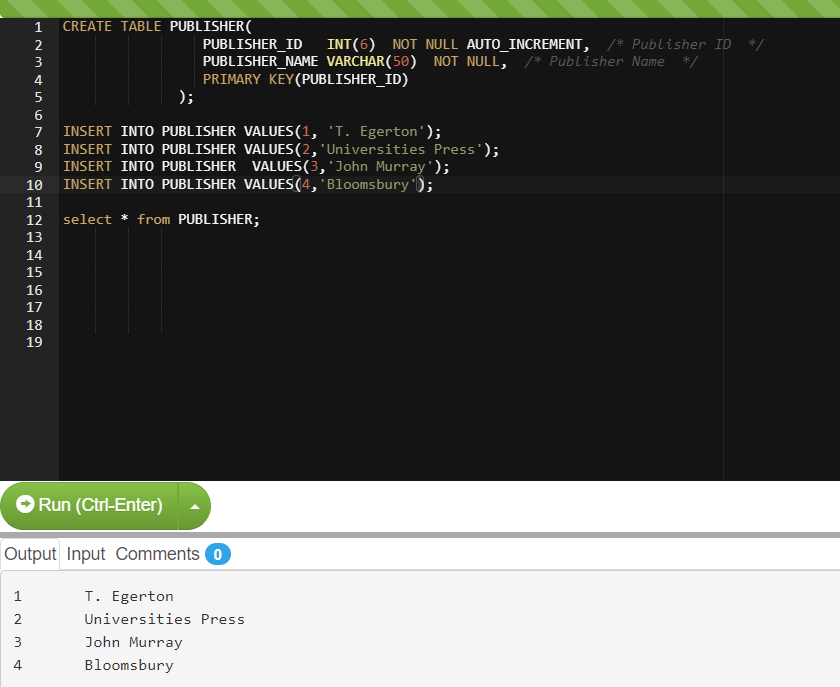
**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO PUBLISHER VALUES(1, 'T. Egerton');

INSERT INTO PUBLISHER VALUES(2,'Universities Press');

INSERT INTO PUBLISHER VALUES(3,'John Murray');

INSERT INTO PUBLISHER VALUES(4,'Bloomsbury');



**4.BOOK**

CREATE TABLE BOOK(

BOOK\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\*Book ID \*/

BOOK\_NAME VARCHAR(50) NOT NULL, /\*Book Name \*/

AUTHOR\_ID INT(6) NOT NULL, /\* Book Author \*/

GENRE\_ID INT(6) NOT NULL, /\* Genre ID \*/

PUBLISHER\_ID INT(6) NOT NULL, /\* Publisher ID \*/

YEAR INT(4) NOT NULL, /\* Year \*/

PRIMARY KEY(BOOK\_ID),

FOREIGN KEY (AUTHOR\_ID)

REFERENCES AUTHOR(AUTHOR\_ID),

FOREIGN KEY (GENRE\_ID)

REFERENCES GENRES(GENRE\_ID),

FOREIGN KEY (PUBLISHER\_ID)

REFERENCES PUBLISHER(PUBLISHER\_ID)

);

**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO BOOK VALUES(1,'Pride and Prejudice', 1, 2, 1,2018);

INSERT INTO BOOK VALUES(2, 'Sense and Sensibility',1,2, 1,2019);

INSERT INTO BOOK VALUES(3, 'Harry Potter and the Philosophers Stone',2,1, 4,2018);

INSERT INTO BOOK VALUES(4, 'Harry Potter and the Goblet of Fire',2,1, 4,2018);

INSERT INTO BOOK VALUES(5, 'Wings of Fire',3,3,2, 2018);

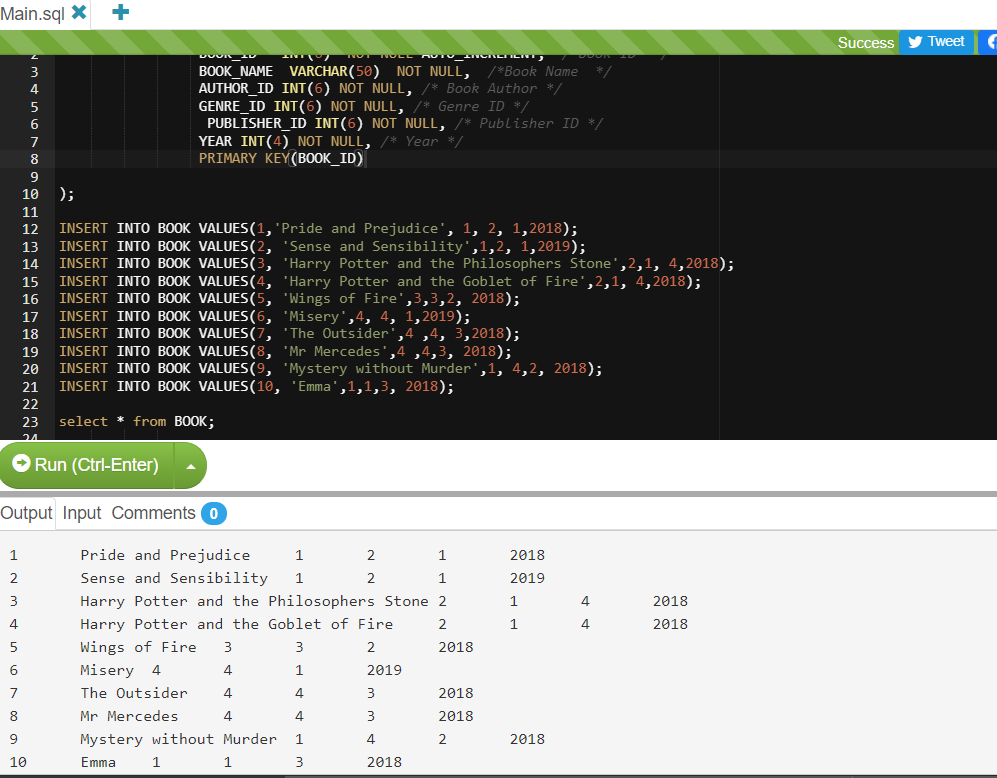
INSERT INTO BOOK VALUES(6, 'Misery',4, 4, 1,2019);

INSERT INTO BOOK VALUES(7, 'The Outsider',4 ,4, 3,2018);

INSERT INTO BOOK VALUES(8, 'Mr Mercedes',4 ,4,3, 2018);

INSERT INTO BOOK VALUES(9, 'Mystery without Murder',1, 4,2, 2018);

INSERT INTO BOOK VALUES(10, 'Emma',1,1,3, 2018);



**5.READER**

CREATE TABLE READER(

READER\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\*Reader ID \*/

READER\_NAME VARCHAR(50) NOT NULL, /\*Reader Name \*/

AGE INT(10) NOT NULL, /\* Age\*/

ADDRESS VARCHAR(50), /\* Reader Address \*/

PRIMARY KEY(READER\_ID)

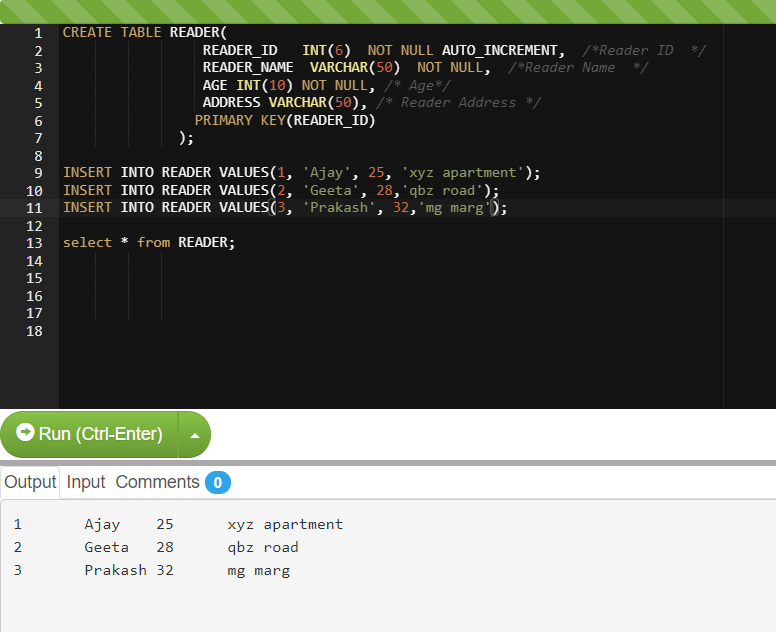
);

**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO READER VALUES(1, 'Ajay', 25, 'xyz apartment');

INSERT INTO READER VALUES(2, 'Geeta', 28,'qbz road');

INSERT INTO READER VALUES(3, 'Prakash', 32,'mg marg');



**6.RATING**

CREATE TABLE RATING(

RATING\_ID INT(6) NOT NULL AUTO\_INCREMENT, /\*Rating ID \*/

RATING INT(2) NOT NULL, /\* Rating \*/

COMMENT VARCHAR(50), /\* Review \*/

READER\_ID INT(6) NOT NULL, /\* Reader ID \*/

BOOK\_ID INT(6) NOT NULL, /\* Book ID \*/

RATING\_DATE DATE NOT NULL, /\* Date\*/

PRIMARY KEY(RATING\_ID),

FOREIGN KEY (READER\_ID)

REFERENCES READER(READER\_ID),

FOREIGN KEY (BOOK\_ID)

REFERENCES BOOK(BOOK\_ID),

CHECK (0<RATING<=10)

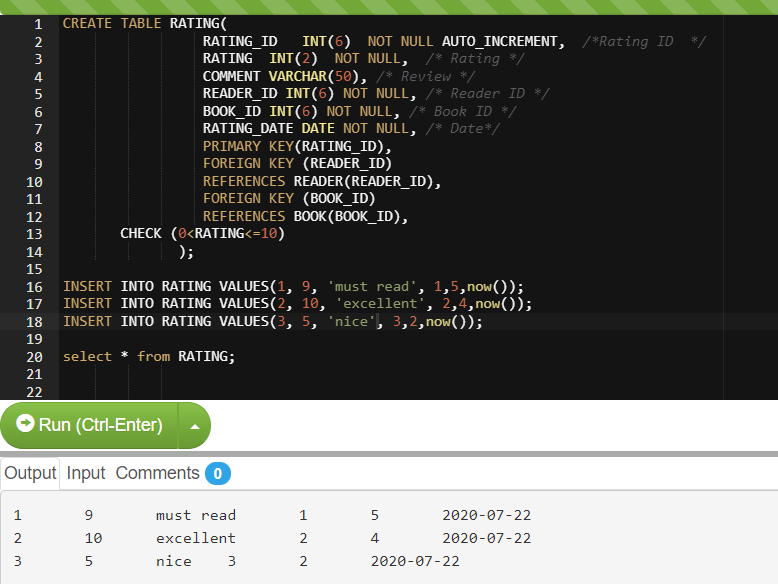
);

**Enter sample data in table (minimum THREE (3) rows per table).**

INSERT INTO RATING VALUES(1, 9, 'must read', 1,5,now());

INSERT INTO RATING VALUES(2, 10, 'excellent', 2,4,now());

INSERT INTO RATING VALUES(3, 5, 'nice', 3,2,now());



## Part B: SQL Statemenṭs

**Question 3: Find the number of books by genre.**

**Answer 3:**

SELECT g.genre\_id, g.genres, count(b.book\_id) no\_of\_books

FROM GENRES AS g

INNER JOIN BOOK AS b on b.genre\_id = g.genre\_id

GROUP BY g.genre\_id, g.genres;

**Output:**

****

**Question 4: Find the top-rated books. Top rated books are defined as the ones with the highest average rating.**

**Answer 4:**

SELECT a.book\_id, a.book\_name, round(avg(b.rating),2) avg\_rating

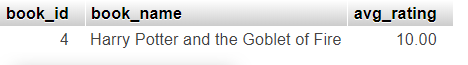
FROM BOOK AS a

LEFT JOIN RATING AS b on b.book\_id = a.book\_id

GROUP BY a.book\_id, a.book\_name

ORDER BY avg\_rating DESC LIMIT 1;

**Output:**



**Question 5: List details of each reader, along with the number of reviews that customer has submitted.**

**Answer 5:**

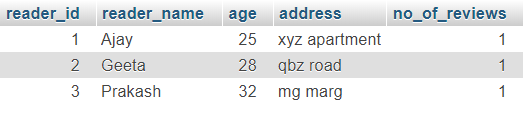
SELECT a.reader\_id, a.reader\_name, a.age, a.address, count(b.reader\_id) no\_of\_reviews

FROM READER AS a

INNER JOIN RATING AS b on b.reader\_id = a.reader\_id

GROUP BY a.reader\_id, a.reader\_name, a.age, a.address;

**Output:**



**Question 6: For each book, list its name, author, publisher, year, along with details of each review. Review details should include: the name of the customer who submitted the review, the date of the review, and the rating given.**

**Answer 6:**

SELECT a.book\_id, a.book\_name, d.author\_name, e.publisher\_name, a.year, c.reader\_name,b.rating, b.rating\_date

FROM BOOK a,

RATING b,

READER c,

AUTHOR d,

PUBLISHER e

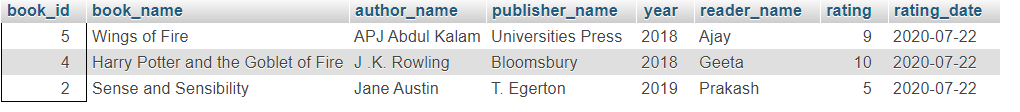
WHERE a.book\_id=b.book\_id

AND b.reader\_id=c.reader\_id

AND a.author\_id=d.author\_id

AND a.publisher\_id=e.publisher\_id;

**Output:**

****

**Question 7: List the top ten books by genre that have a rating of 8 or more, published between 2018 and 2019, with at least 10 reviews.**

**Answer 7:**

SELECT c.genres, a.book\_name, count(b.BOOK\_ID)

FROM BOOK a,

RATING b,

GENRES c

WHERE a.book\_id=b.book\_id

AND a.genre\_id=c.genre\_id

AND b.rating >=8

AND a.year between 2018 and 2019

having count(b.BOOK\_ID)>=10;

**Output:**

****

## Conclusion

In this assignment, I have created an ‘Online Bookstore System’ that creates and saves new books and stores data of author, publisher and reader efficiently. All data are stored in 3NF form which leads to no data duplication and data inconsistency. Using MySQL DBMS, data can be easily modified using SQL DML statements that includes (SELECT, INSERT, UPDATE, DELETE). DBMS has made life easier by storing data efficiently. Paper work has come to an end. This system is tested with dummy data and is working fine. Searching for highest rated book has become quite easy and faster through this system. The records are easily accessible for any book, reader, author or publisher. DBMS also keeps backup, thus in case of failure data are recovered easily without any loss. Thus resulting into an efficient system to record details of books of any genre.