Indian Institute of Technology Gandhinagar

Library Management System

CS 432 : Project Report

Group - x

Team Members

Jayesh Bhadange	20110082
Pranav Rathod	20110143
Naval Jaggi	20110118
Harendra Khatik	20110072
Sparsh Dawra	20110203
Manpreet Singh	20110109
Sidharth Joshi	19110169
Gajanan Donge	20110061
Manish Jangir	20110107
Sankarshan	20110184
Riya Dhantoliya	20110168
Pintu Kumar Meena	19110193

Under the guidance of

Prof. Mayank Singh

CS 432: Databases

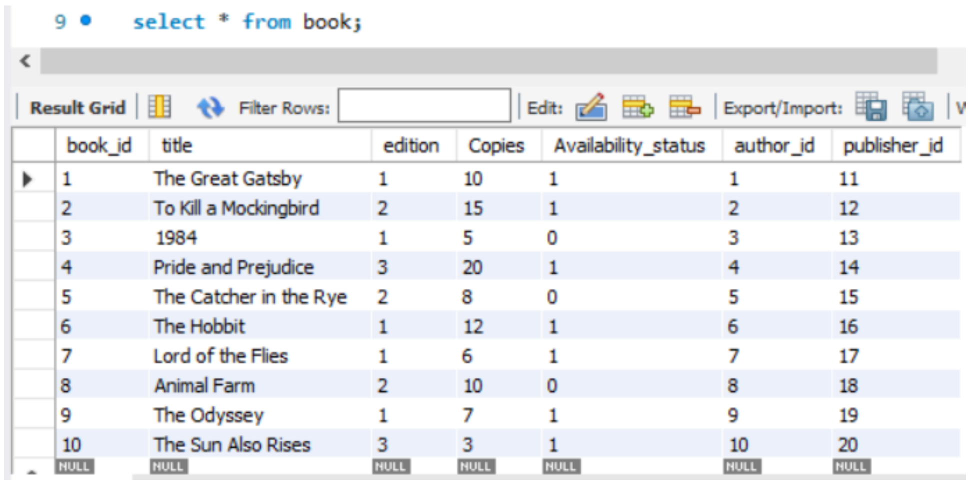
Assignment 2: DEVELOPING THE DBMS

Responsibility of G1

Part 2

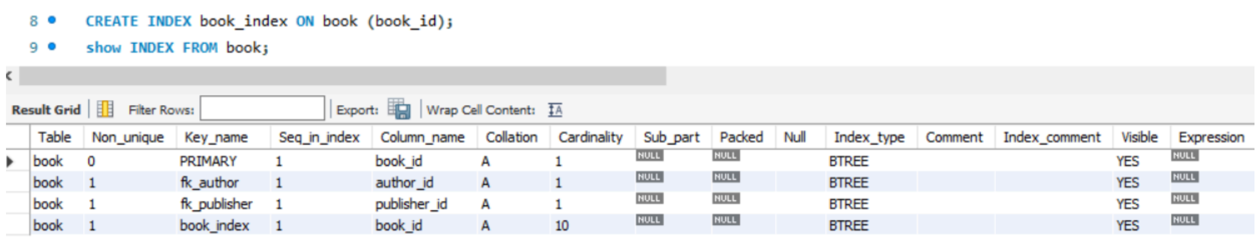
- **Indexing:**

Indexing is an important method in dbms which improves the efficiency of many operations. Indexing helps in faster data retrieval and manipulation operations. Indexing creates a map that links the values in a column to the corresponding rows in the table. This enables DBMS to quickly locate and retrieve the data that matches specific search criteria, reducing the amount of time and resources required to retrieve the data which also reduces the disk space.



```
select * from book;
```

book_id	title	edition	Copies	Availability_status	author_id	publisher_id
1	The Great Gatsby	1	10	1	1	11
2	To Kill a Mockingbird	2	15	1	2	12
3	1984	1	5	0	3	13
4	Pride and Prejudice	3	20	1	4	14
5	The Catcher in the Rye	2	8	0	5	15
6	The Hobbit	1	12	1	6	16
7	Lord of the Flies	1	6	1	7	17
8	Animal Farm	2	10	0	8	18
9	The Odyssey	1	7	1	9	19
10	The Sun Also Rises	3	3	1	10	20
NULL	NULL	NULL	NULL	NULL	NULL	NULL



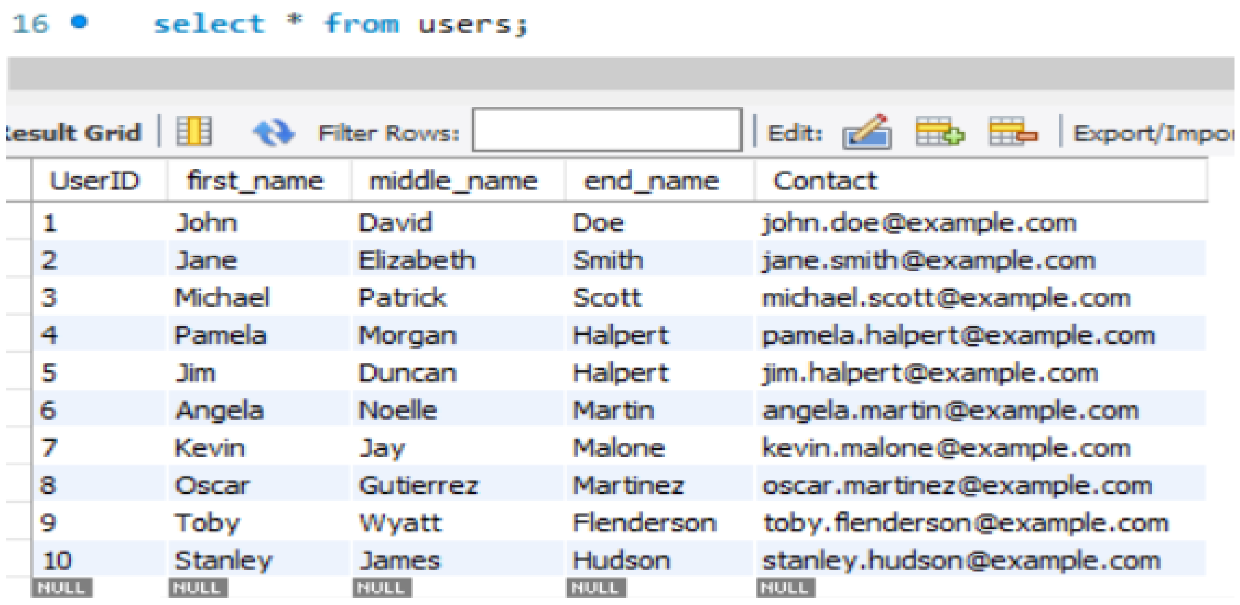
```sql
CREATE INDEX book_index ON book (book_id);
show INDEX FROM book;
```

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
book	0	PRIMARY	1	book_id	A	1	NULL	NULL		BTREE			YES	NULL
book	1	fk_author	1	author_id	A	1	NULL	NULL		BTREE			YES	NULL
book	1	fk_publisher	1	publisher_id	A	1	NULL	NULL		BTREE			YES	NULL
book	1	book_index	1	book_id	A	10	NULL	NULL		BTREE			YES	NULL

- ## Table extension:

In library database there is schema named as 'users' which stores information about students or faculty (name and email id)

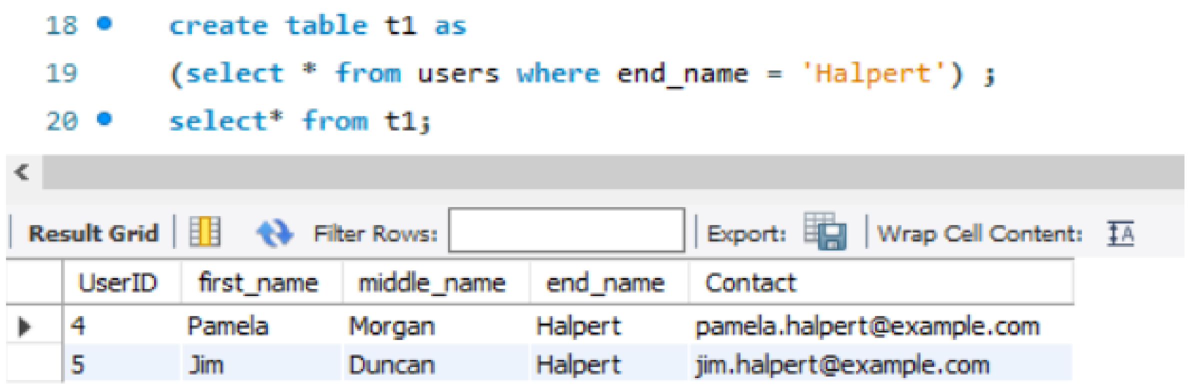


```sql
select * from users;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

UserID	first_name	middle_name	end_name	Contact
1	John	David	Doe	john.doe@example.com
2	Jane	Elizabeth	Smith	jane.smith@example.com
3	Michael	Patrick	Scott	michael.scott@example.com
4	Pamela	Morgan	Halpert	pamela.halpert@example.com
5	Jim	Duncan	Halpert	jim.halpert@example.com
6	Angela	Noelle	Martin	angela.martin@example.com
7	Kevin	Jay	Malone	kevin.malone@example.com
8	Oscar	Gutierrez	Martinez	oscar.martinez@example.com
9	Toby	Wyatt	Flenderson	toby.flenderson@example.com
10	Stanley	James	Hudson	stanley.hudson@example.com
NULL	NULL	NULL	NULL	NULL

Creation of table that have the same schema as an existing table



```sql
create table t1 as
(select * from users where end_name = 'Halpert') ;
select* from t1;
```

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

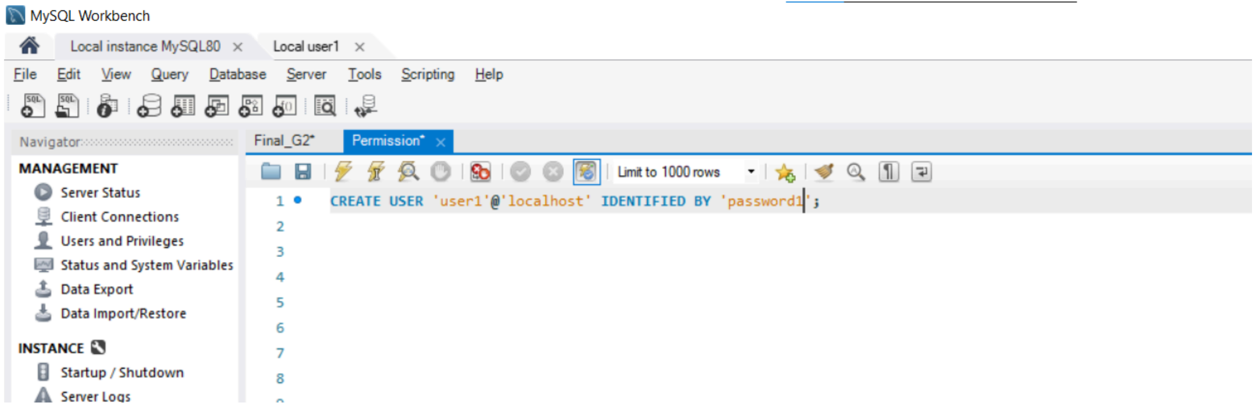
UserID	first_name	middle_name	end_name	Contact
4	Pamela	Morgan	Halpert	pamela.halpert@example.com
5	Jim	Duncan	Halpert	jim.halpert@example.com

User-defined data type: User-defined data types are custom data types that are created by a user in a database management system (DBMS) to represent specific types of data.

Responsibilities of G2:-

Question 1.1

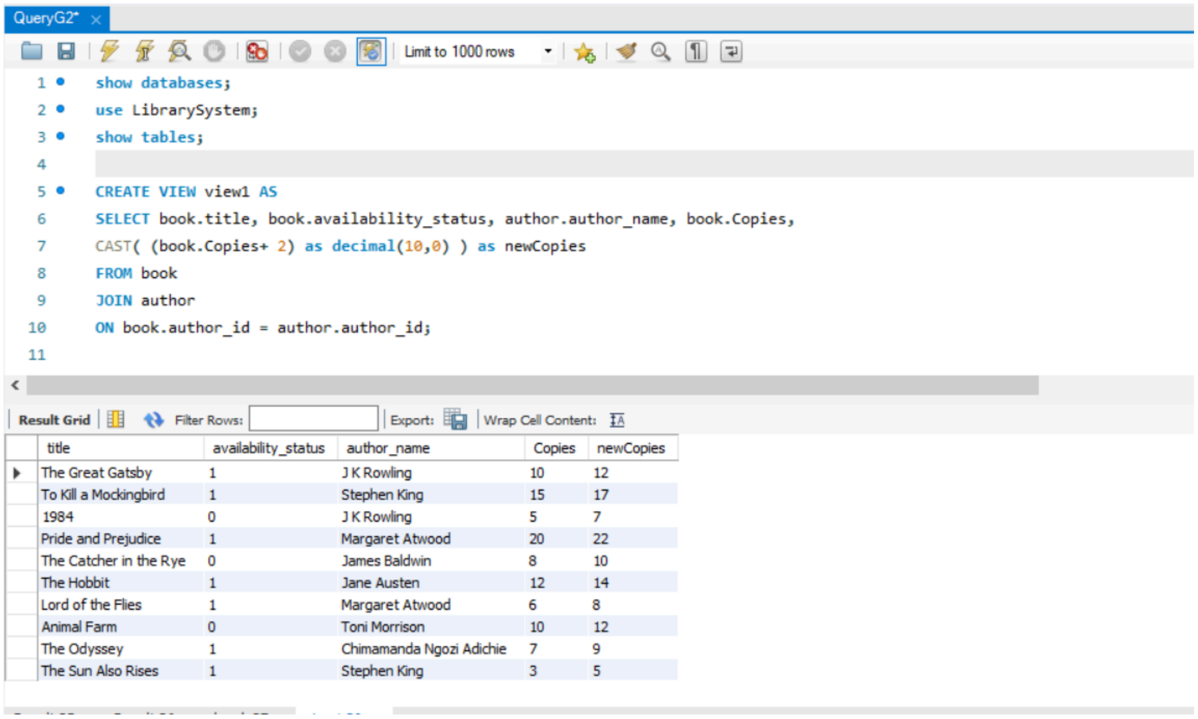
We have created a user1 with password1.



```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
```

Question 1.2

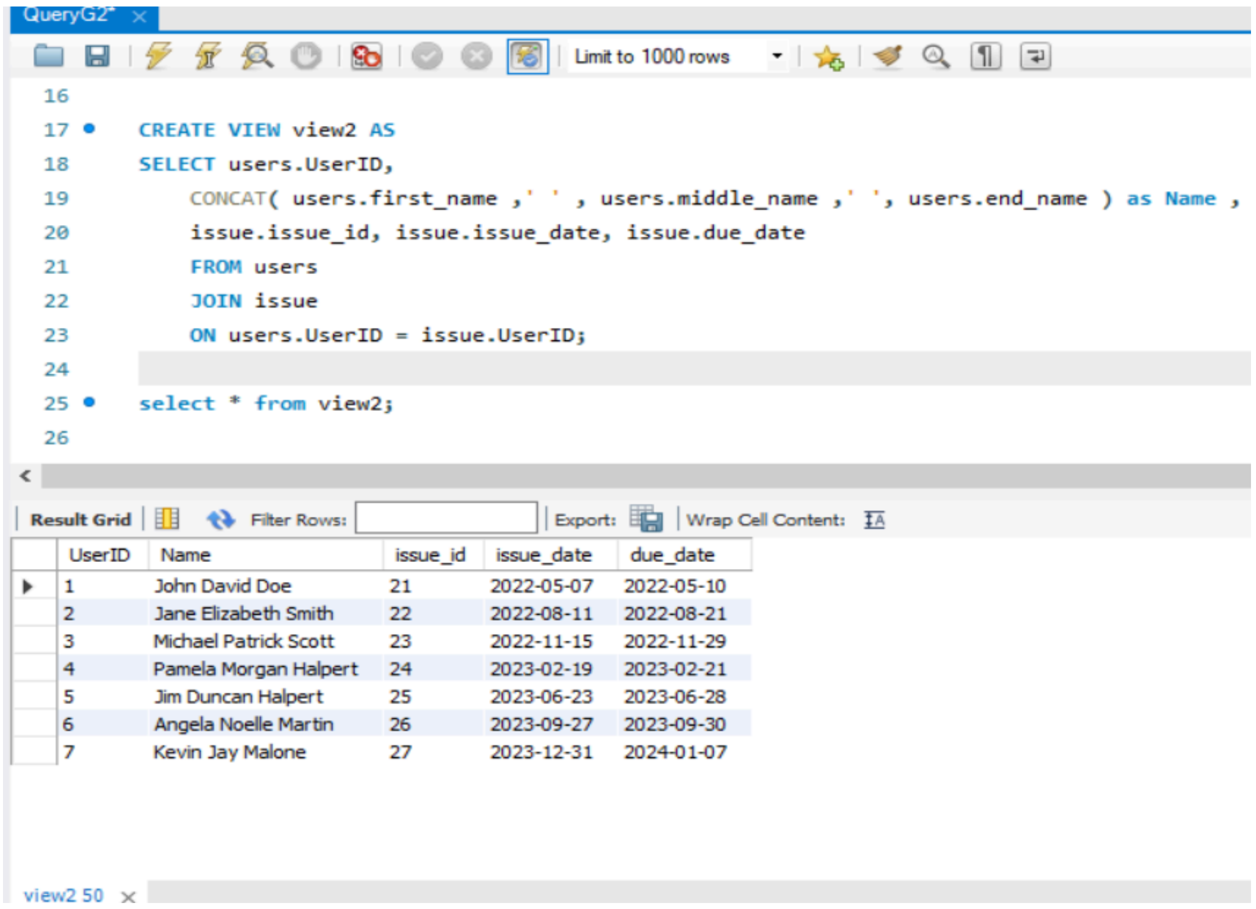
Created view1 from table book and author. Added one additional column of updated Copies of book (i.e newCopies).



```
show databases;
use LibrarySystem;
show tables;

CREATE VIEW view1 AS
SELECT book.title, book.availability_status, author.author_name, book.Copies,
CAST( (book.Copies+ 2) as decimal(10,0) ) as newCopies
FROM book
JOIN author
ON book.author_id = author.author_id;
```

title	availability_status	author_name	Copies	newCopies
The Great Gatsby	1	J K Rowling	10	12
To Kill a Mockingbird	1	Stephen King	15	17
1984	0	J K Rowling	5	7
Pride and Prejudice	1	Margaret Atwood	20	22
The Catcher in the Rye	0	James Baldwin	8	10
The Hobbit	1	Jane Austen	12	14
Lord of the Flies	1	Margaret Atwood	6	8
Animal Farm	0	Toni Morrison	10	12
The Odyssey	1	Chimamanda Ngozi Adichie	7	9
The Sun Also Rises	1	Stephen King	3	5

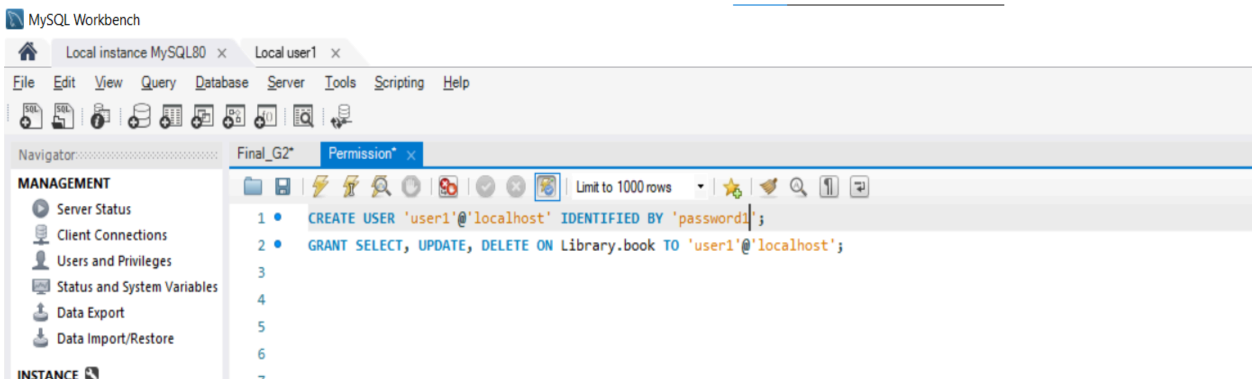
Created view2 from table users and issue. Added one additional column of full name instead of three columns of first name, middle name and last name (i.e Name).



```
CREATE VIEW view2 AS
SELECT users.UserID,
    CONCAT( users.first_name ,' ' , users.middle_name ,' ', users.end_name ) as Name ,
    issue.issue_id, issue.issue_date, issue.due_date
    FROM users
    JOIN issue
    ON users.UserID = issue.UserID;

select * from view2;
```

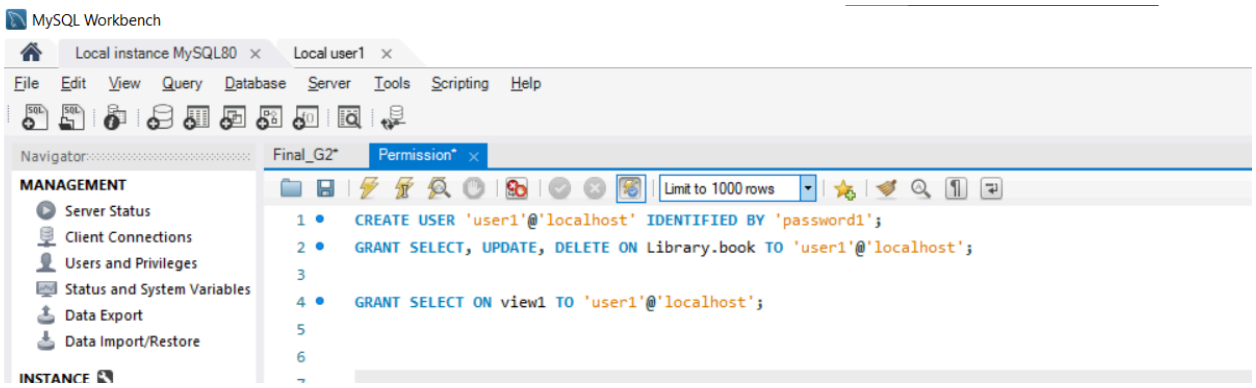
UserID	Name	issue_id	issue_date	due_date
1	John David Doe	21	2022-05-07	2022-05-10
2	Jane Elizabeth Smith	22	2022-08-11	2022-08-21
3	Michael Patrick Scott	23	2022-11-15	2022-11-29
4	Pamela Morgan Halpert	24	2023-02-19	2023-02-21
5	Jim Duncan Halpert	25	2023-06-23	2023-06-28
6	Angela Noelle Martin	26	2023-09-27	2023-09-30
7	Kevin Jay Malone	27	2023-12-31	2024-01-07

Question 1.3



```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
GRANT SELECT, UPDATE, DELETE ON Library.book TO 'user1'@'localhost';
```

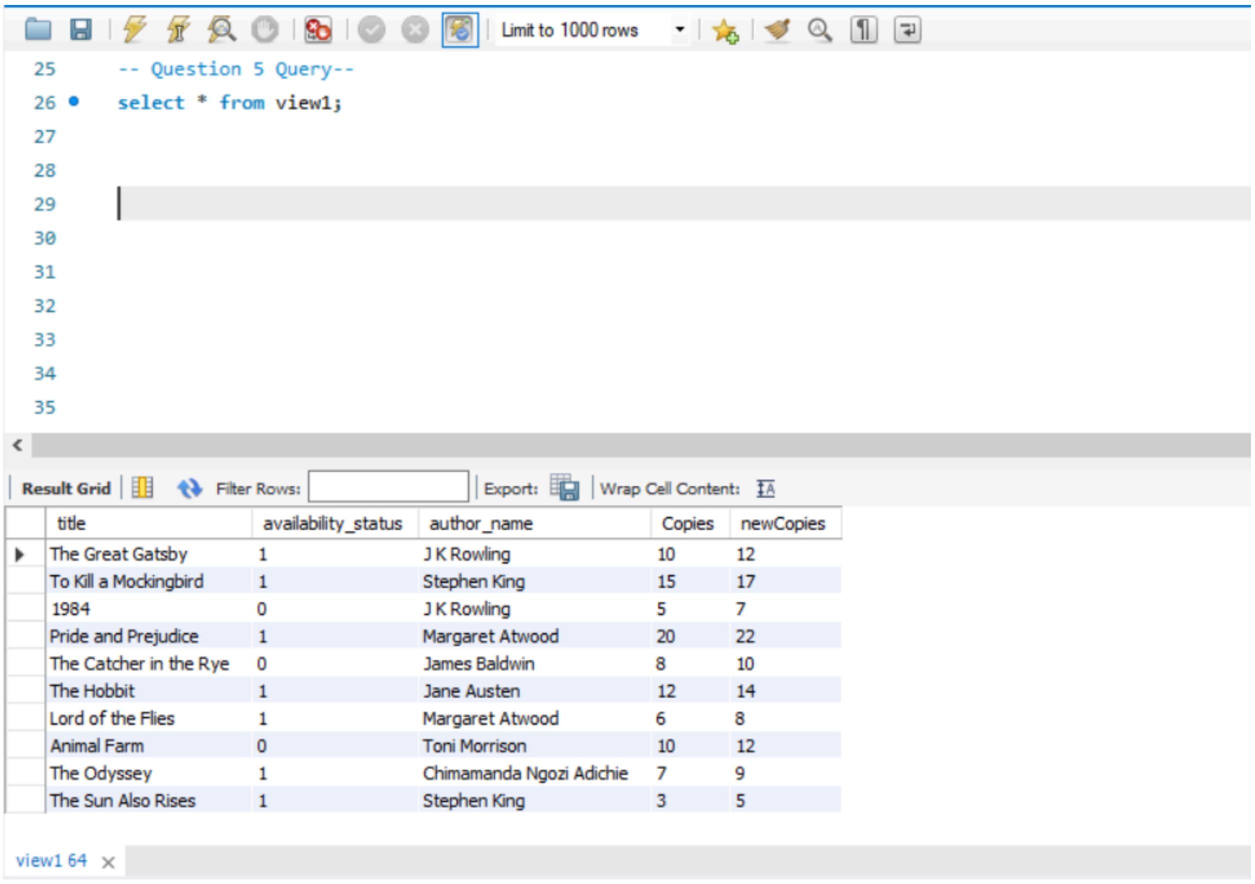
Question 1.4



```sql
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
GRANT SELECT, UPDATE, DELETE ON Library.book TO 'user1'@'localhost';

GRANT SELECT ON view1 TO 'user1'@'localhost';
```

Question 1.5

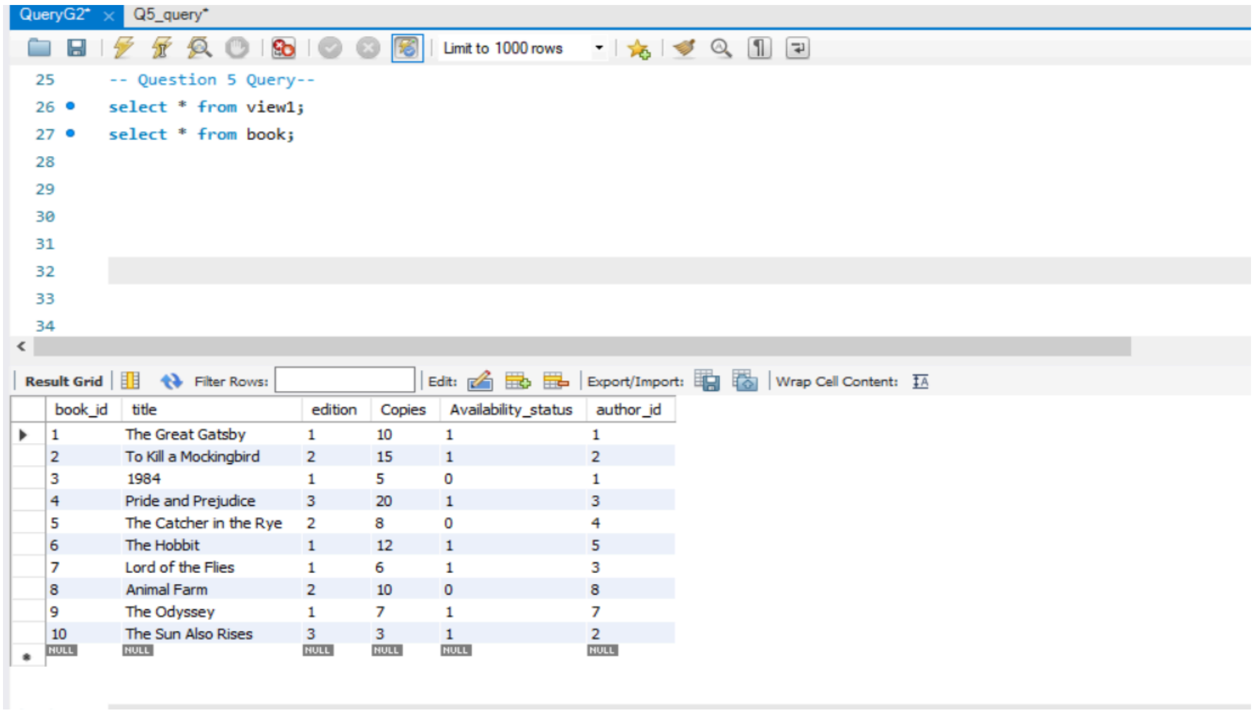
Select operation on view1 as user1



```sql
-- Question 5 Query--
select * from view1;
```

title	availability_status	author_name	Copies	newCopies
The Great Gatsby	1	J K Rowling	10	12
To Kill a Mockingbird	1	Stephen King	15	17
1984	0	J K Rowling	5	7
Pride and Prejudice	1	Margaret Atwood	20	22
The Catcher in the Rye	0	James Baldwin	8	10
The Hobbit	1	Jane Austen	12	14
Lord of the Flies	1	Margaret Atwood	6	8
Animal Farm	0	Toni Morrison	10	12
The Odyssey	1	Chimamanda Ngozi Adichie	7	9
The Sun Also Rises	1	Stephen King	3	5

Select operation on book as user1


QueryG2* × Q5_query*

Limit to 1000 rows

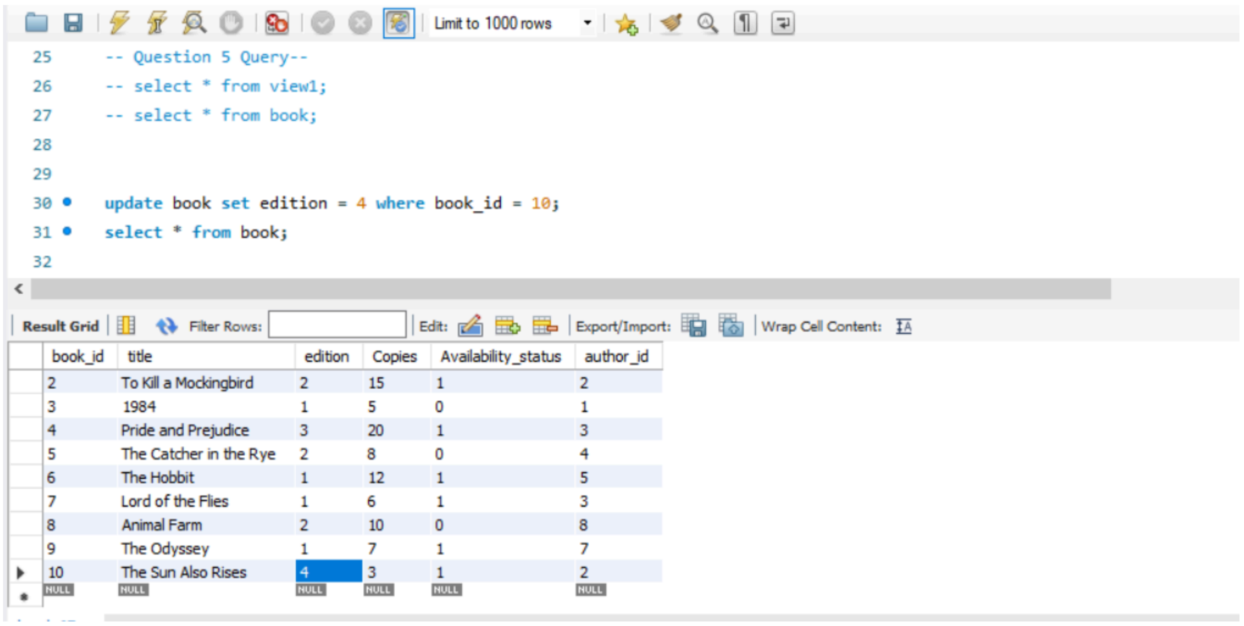
```
-- Question 5 Query--
select * from view1;
select * from book;
```

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

book_id	title	edition	Copies	Availability_status	author_id
1	The Great Gatsby	1	10	1	1
2	To Kill a Mockingbird	2	15	1	2
3	1984	1	5	0	1
4	Pride and Prejudice	3	20	1	3
5	The Catcher in the Rye	2	8	0	4
6	The Hobbit	1	12	1	5
7	Lord of the Flies	1	6	1	3
8	Animal Farm	2	10	0	8
9	The Odyssey	1	7	1	7
10	The Sun Also Rises	3	3	1	2
NULL	NULL	NULL	NULL	NULL	NULL

User1 has been granted the SELECT permission for both book and view. That's Why the SELECT operation didn't throw any error.

Update operation on book as user1


Limit to 1000 rows

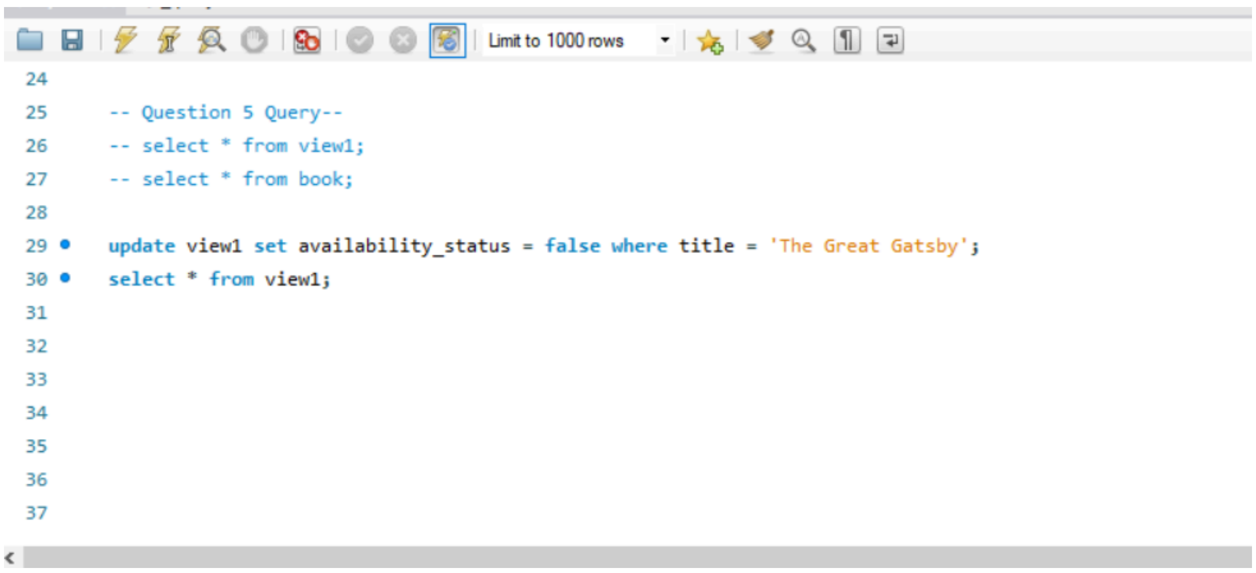
```
-- Question 5 Query--
-- select * from view1;
-- select * from book;


update book set edition = 4 where book_id = 10;
select * from book;
```

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

book_id	title	edition	Copies	Availability_status	author_id
2	To Kill a Mockingbird	2	15	1	2
3	1984	1	5	0	1
4	Pride and Prejudice	3	20	1	3
5	The Catcher in the Rye	2	8	0	4
6	The Hobbit	1	12	1	5
7	Lord of the Flies	1	6	1	3
8	Animal Farm	2	10	0	8
9	The Odyssey	1	7	1	7
10	The Sun Also Rises	4	3	1	2
NULL	NULL	NULL	NULL	NULL	NULL

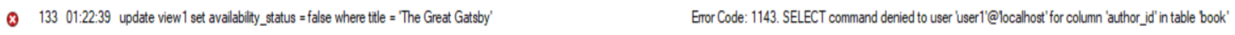
In Book relation we updated the value of book edition at book_id number 10, from 3 to 4. Error didn’t occur because we have given UPDATE permission for book to user1.

Update operation on view1 as user1:



```

-- Question 5 Query--
-- select * from view1;
-- select * from book;

update view1 set availability_status = false where title = 'The Great Gatsby';
select * from view1;
```

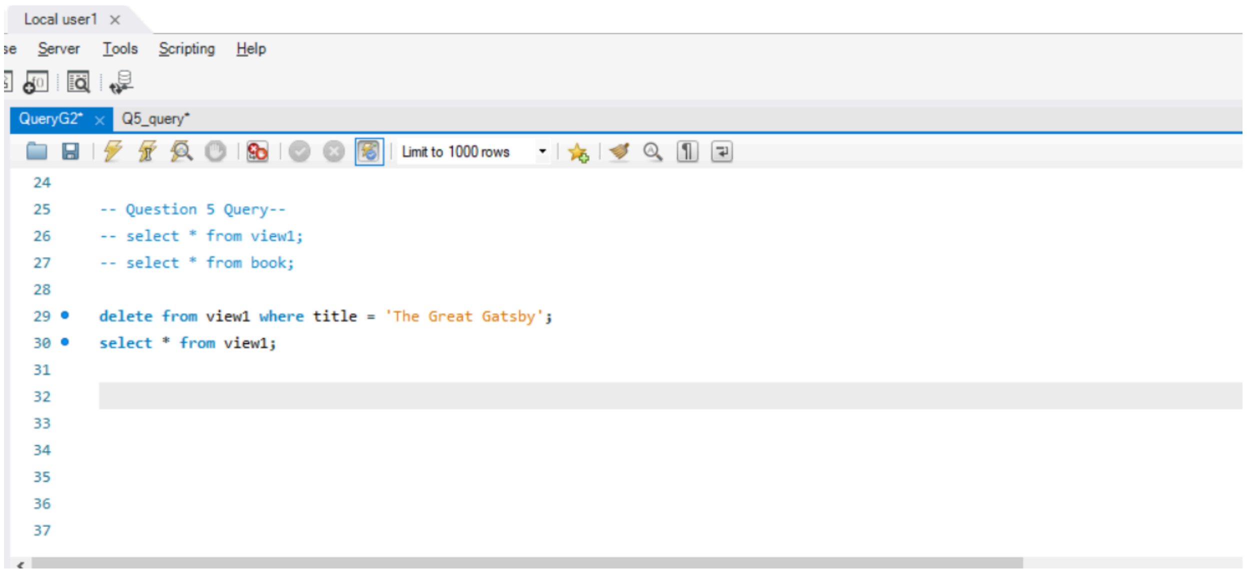
Output



```
133  01:22:39  update view1 set availability_status = false where title = 'The Great Gatsby'     Error Code: 1143. SELECT command denied to user 'user1'@'localhost' for column 'author_id' in table 'book'
```

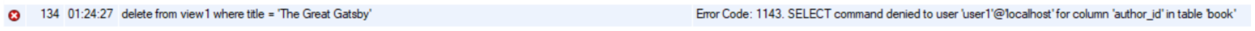
Error occurred because we have not given UPDATE permission of view1 to user1.

Delete operation on view1 as user1



```

-- Question 5 Query--
-- select * from view1;
-- select * from book;

delete from view1 where title = 'The Great Gatsby';
select * from view1;
```

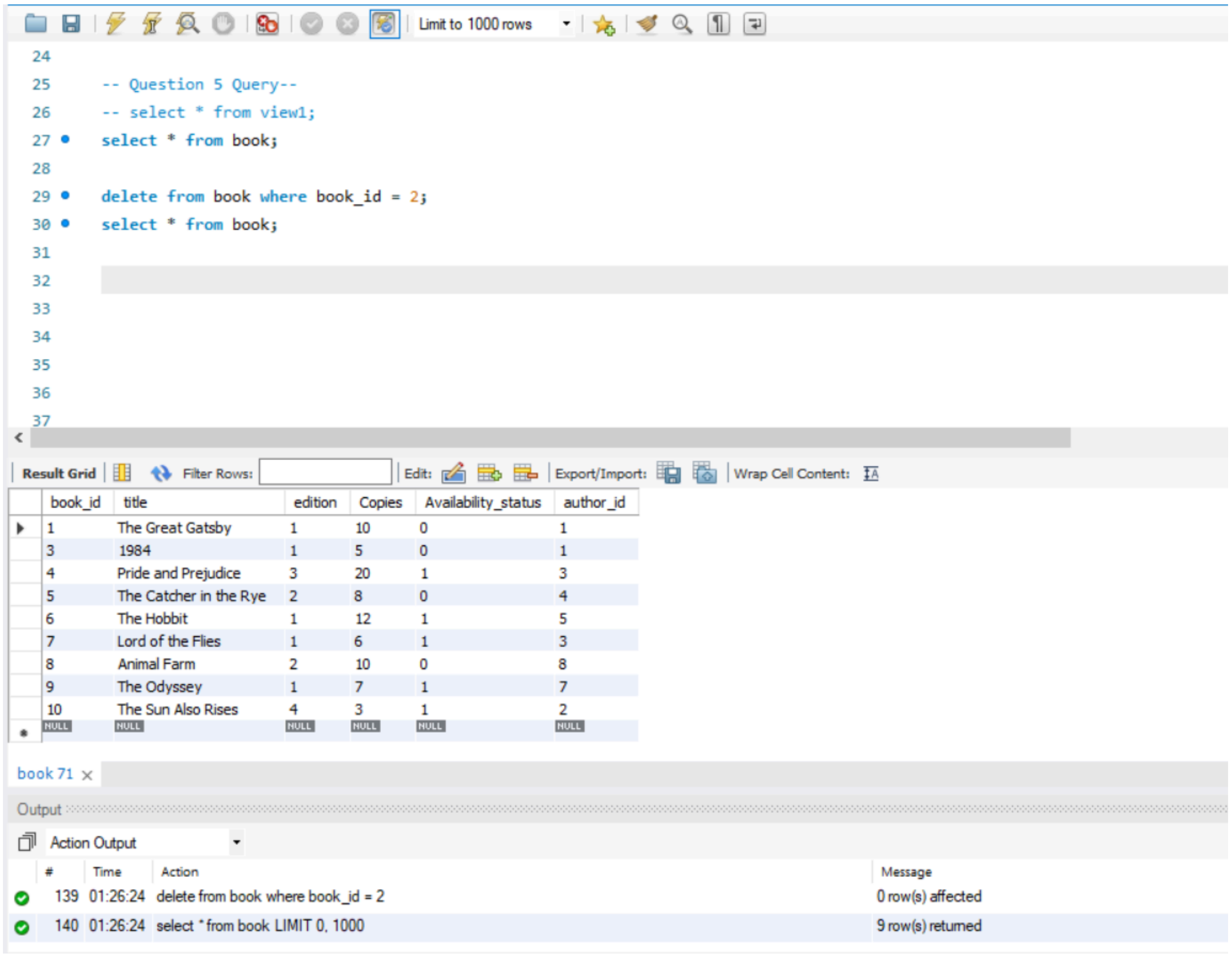
Output



```
134  01:24:27  delete from view1 where title = 'The Great Gatsby'    Error Code: 1143. SELECT command denied to user 'user1'@'localhost' for column 'author_id' in table 'book'
```

Error occurred because we have not given UPDATE permission of view1 to user1.

Delete operation on book table as user1



Limit to 1000 rows

```sql
-- Question 5 Query--
-- select * from view1;
select * from book;

delete from book where book_id = 2;
select * from book;
```

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

book_id	title	edition	Copies	Availability_status	author_id
1	The Great Gatsby	1	10	0	1
3	1984	1	5	0	1
4	Pride and Prejudice	3	20	1	3
5	The Catcher in the Rye	2	8	0	4
6	The Hobbit	1	12	1	5
7	Lord of the Flies	1	6	1	3
8	Animal Farm	2	10	0	8
9	The Odyssey	1	7	1	7
10	The Sun Also Rises	4	3	1	2
NULL	NULL	NULL	NULL	NULL	NULL

book 71 ×

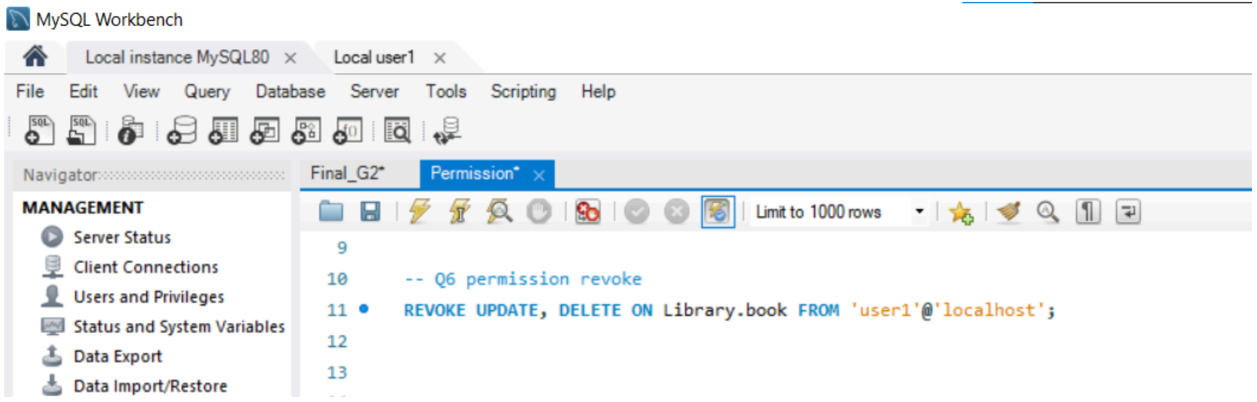
Output

Action Output

```
#    Time      Action                                  Message
139  01:26:24  delete from book where book_id = 2      0 row(s) affected
140  01:26:24  select * from book LIMIT 0, 1000        9 row(s) returned
```

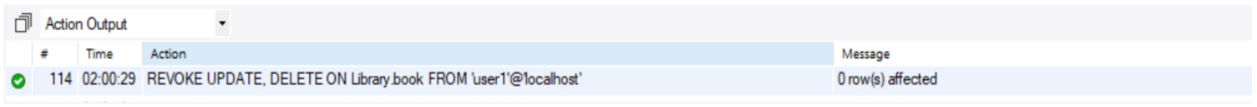
As we can see in the table above, we deleted the row with book_id number 2. Here error didn’t occur because we have given DELETE permission for book to user1.

Question 1.6



```sql
-- Q6 permission revoke
REVOKE UPDATE, DELETE ON Library.book FROM 'user1'@'localhost';
```

Output

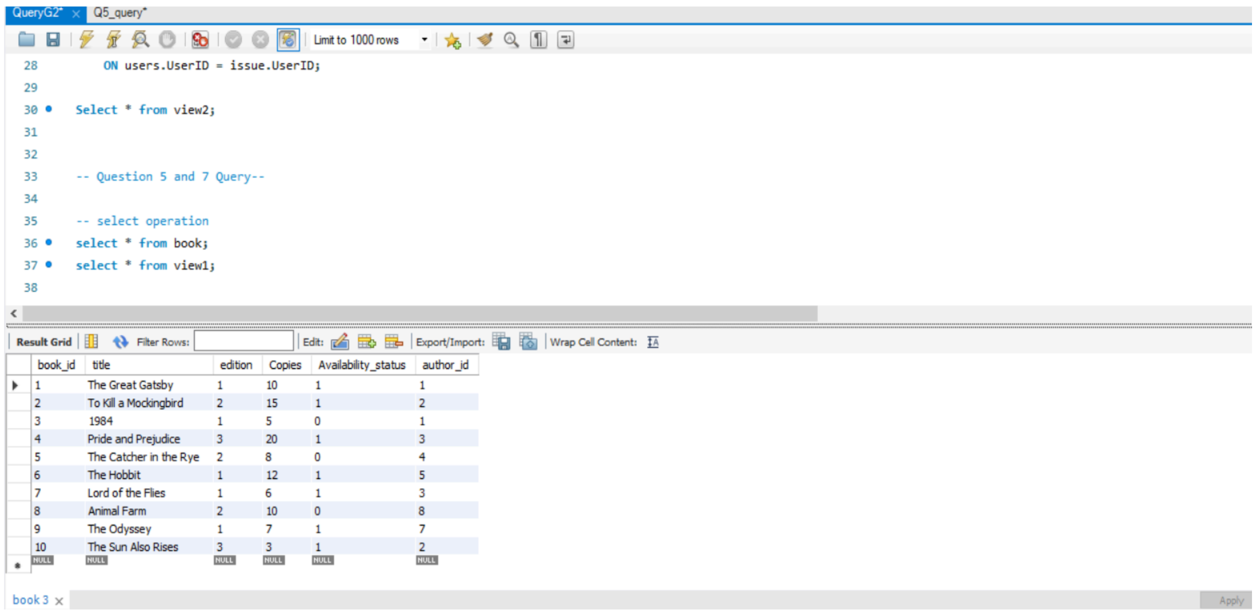


```
#    Time      Action                                                     Message
114  02:00:29  REVOKE UPDATE, DELETE ON Library.book FROM 'user1'@'localhost'  0 row(s) affected
```

Now, user1 only has SELECT permission for book and view1. From now, user1 can neither update nor delete any entry in book and view1. User1 is a viewer from now-onwards.

Question 1.7

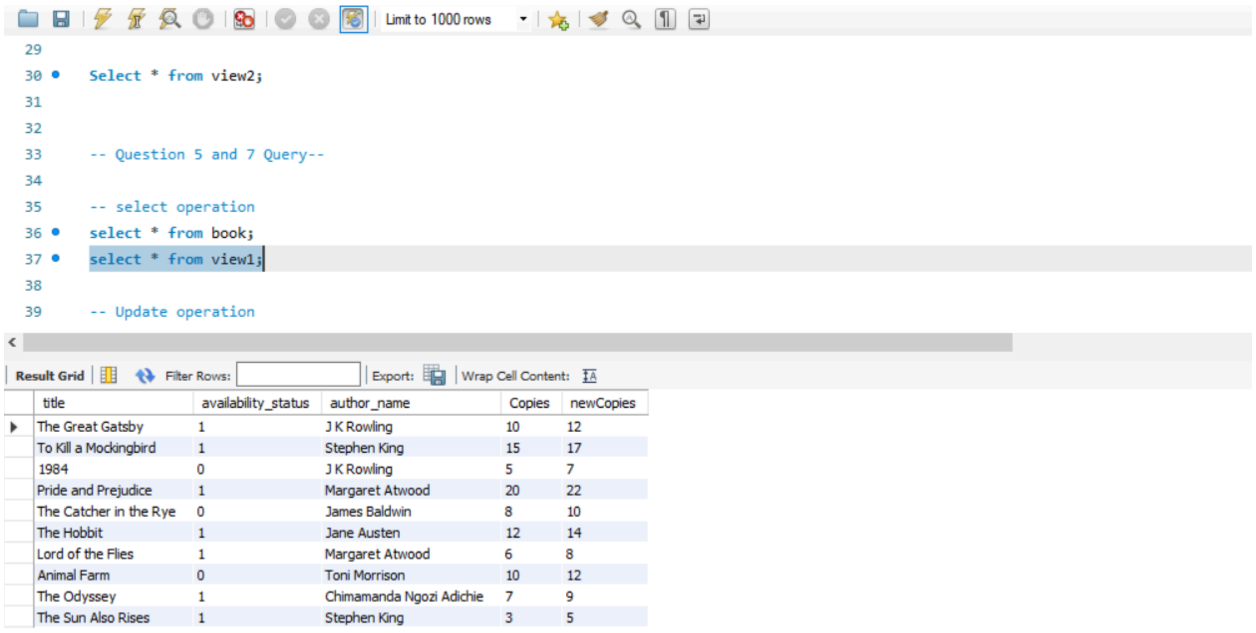
SELECT operation on book ->



```sql
        ON users.UserID = issue.UserID;

    Select * from view2;


    -- Question 5 and 7 Query--

    -- select operation
    select * from book;
    select * from view1;
```

book_id	title	edition	Copies	Availability_status	author_id
1	The Great Gatsby	1	10	1	1
2	To Kill a Mockingbird	2	15	1	2
3	1984	1	5	0	1
4	Pride and Prejudice	3	20	1	3
5	The Catcher in the Rye	2	8	0	4
6	The Hobbit	1	12	1	5
7	Lord of the Flies	1	6	1	3
8	Animal Farm	2	10	0	8
9	The Odyssey	1	7	1	7
10	The Sun Also Rises	3	3	1	2
NULL	NULL	NULL	NULL	NULL	NULL

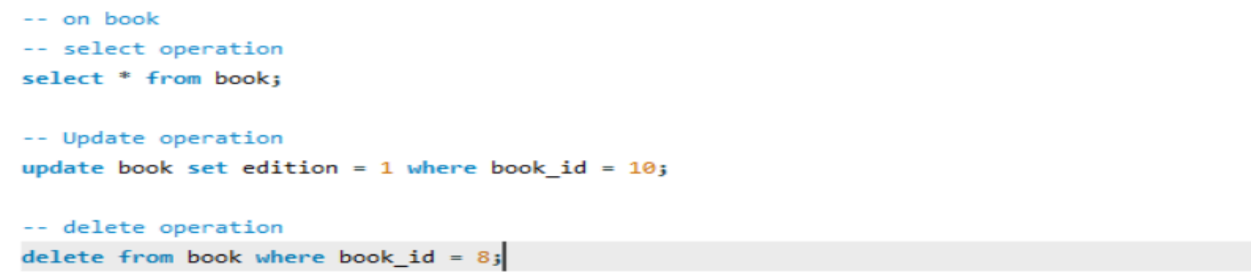
SELECT operation on view1 ->



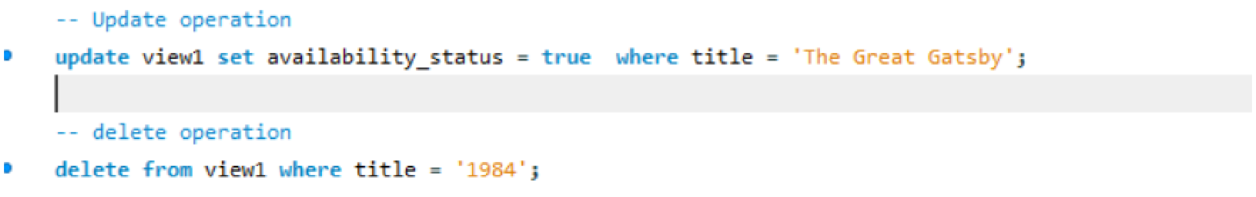
```
Select * from view2;


-- Question 5 and 7 Query--

-- select operation
select * from book;
select * from view1;

-- Update operation
```

title	availability_status	author_name	Copies	newCopies
The Great Gatsby	1	J K Rowling	10	12
To Kill a Mockingbird	1	Stephen King	15	17
1984	0	J K Rowling	5	7
Pride and Prejudice	1	Margaret Atwood	20	22
The Catcher in the Rye	0	James Baldwin	8	10
The Hobbit	1	Jane Austen	12	14
Lord of the Flies	1	Margaret Atwood	6	8
Animal Farm	0	Toni Morrison	10	12
The Odyssey	1	Chimamanda Ngozi Adichie	7	9
The Sun Also Rises	1	Stephen King	3	5

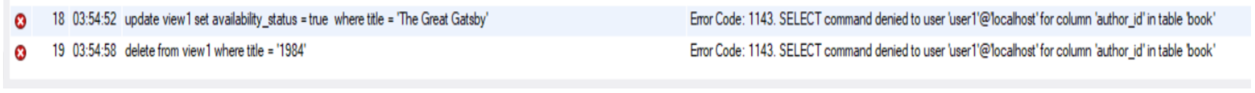
UPDATE and DELETE operation on view1 and book ->



```
-- on book
-- select operation
select * from book;

-- Update operation
update book set edition = 1 where book_id = 10;

-- delete operation
delete from book where book_id = 8;
```



```
-- Update operation
update view1 set availability_status = true  where title = 'The Great Gatsby';

-- delete operation
delete from view1 where title = '1984';
```



```
18  03:54:52  update view1 set availability_status = true  where title = 'The Great Gatsby'   Error Code: 1143. SELECT command denied to user 'user1'@'localhost' for column 'author_id' in table 'book'
19  03:54:58  delete from view1 where title = '1984'                                        Error Code: 1143. SELECT command denied to user 'user1'@'localhost' for column 'author_id' in table 'book'
```

Error occurred because both view1 and book have not been granted UPDATE and SELECT permission.

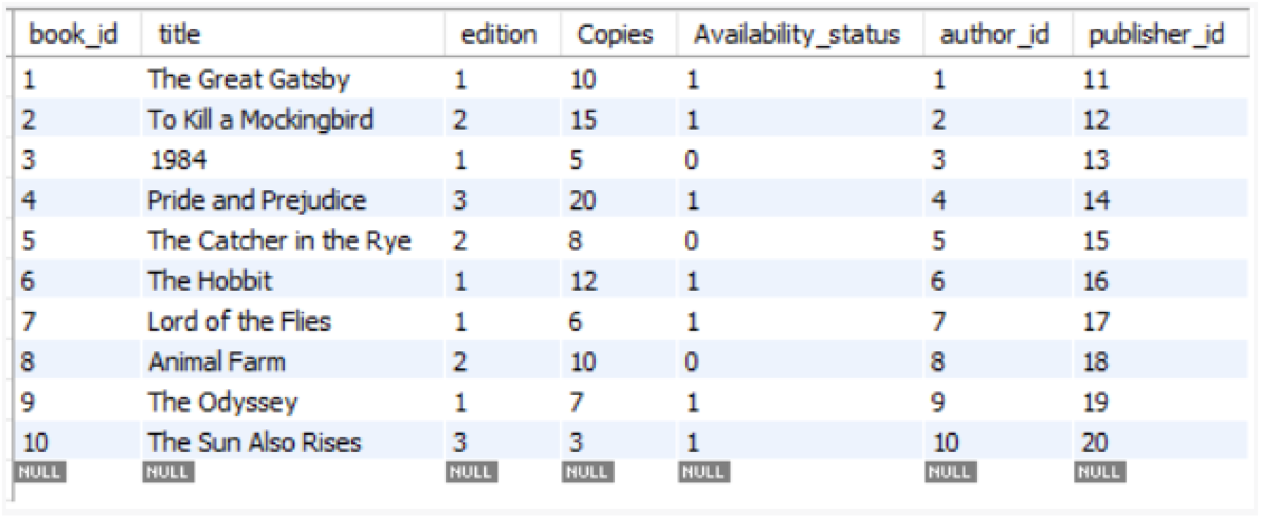
<u>Question 2:-</u>

Referential integrity violation is a type of error that occurs in a database when a foreign key constraint is violated. A foreign key is a field or set of fields in a table that refers to the primary key of another table, establishing a link between the two tables. The foreign key constraint ensures that the values in the foreign key column of the referencing table must match the values in the primary key column of the referenced table.

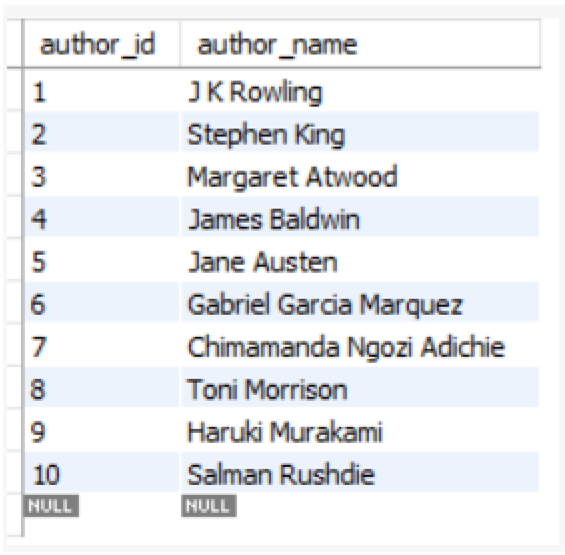
There are several causes of referential integrity violation in a database, including:
Inserting or updating records in the referencing table with foreign key values that do not exist in the referenced table.
Deleting records from the referenced table without updating or deleting the corresponding records in the referencing table.
Updating the primary key value in the referenced table without updating the corresponding foreign key values in the referencing table

Here, we are using the book-publisher and book-author relationship to show the referential integrity violation:

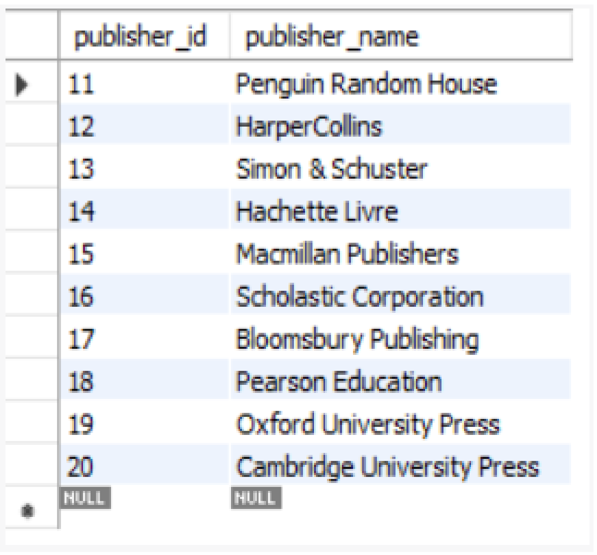
Below are the table without any update/ deletion:



book_id	title	edition	Copies	Availability_status	author_id	publisher_id
1	The Great Gatsby	1	10	1	1	11
2	To Kill a Mockingbird	2	15	1	2	12
3	1984	1	5	0	3	13
4	Pride and Prejudice	3	20	1	4	14
5	The Catcher in the Rye	2	8	0	5	15
6	The Hobbit	1	12	1	6	16
7	Lord of the Flies	1	6	1	7	17
8	Animal Farm	2	10	0	8	18
9	The Odyssey	1	7	1	9	19
10	The Sun Also Rises	3	3	1	10	20
NULL	NULL	NULL	NULL	NULL	NULL	NULL

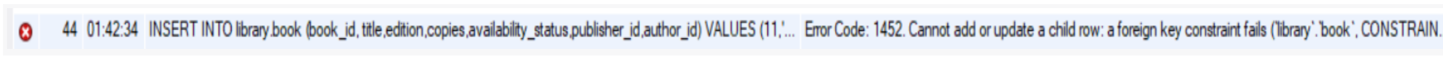


author_id	author_name
1	J K Rowling
2	Stephen King
3	Margaret Atwood
4	James Baldwin
5	Jane Austen
6	Gabriel Garcia Marquez
7	Chimamanda Ngozi Adichie
8	Toni Morrison
9	Haruki Murakami
10	Salman Rushdie
NULL	NULL



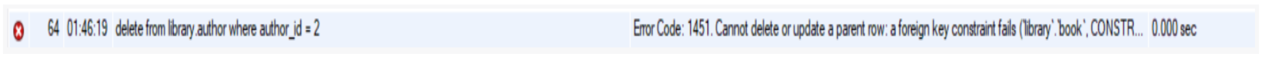
publisher_id	publisher_name
11	Penguin Random House
12	HarperCollins
13	Simon & Schuster
14	Hachette Livre
15	Macmillan Publishers
16	Scholastic Corporation
17	Bloomsbury Publishing
18	Pearson Education
19	Oxford University Press
20	Cambridge University Press
NULL	NULL

When we apply the insertion operation in the book table with the author_id that is not in the author table then the referential integrity violation occurs.



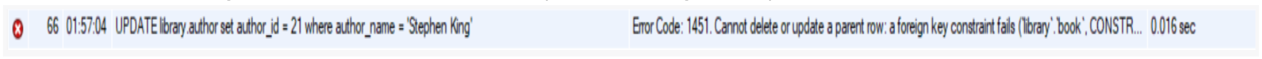
```
44 01:42:34 INSERT INTO library.book (book_id, title,edition,copies,availability_status,publisher_id,author_id) VALUES (11,'... Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('library'.'book', CONSTRAIN.
```

When we apply the delete operation in the author table(referenced table), then also the violation occurs as mentioned in the points above.



```
64 01:46:19 delete from library.author where author_id = 2     Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('library'.'book', CONSTR... 0.000 sec
```

When we update the value of author_id in the author table(referenced table) without updating the corresponding value in the book table(referencing table), then also the violation occurs.



```
66 01:57:04 UPDATE library.author set author_id = 21 where author_name = 'Stephen King'     Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('library'.'book', CONSTR... 0.016 sec
```

When a referential integrity violation occurs when updating a tuple in a referenced table, there are a few ways to handle it depending on the specific requirements of the database and application. Here are a few common strategies:

1) Prevent the update: This is the default behavior of most database management systems. When a referential integrity violation occurs, the DBMS will prevent the update from occurring and return an error message. This ensures that the database remains consistent and that foreign key relationships are maintained.
2) Cascade updates: With cascade updates, the DBMS will automatically update any dependent tuples in the referencing table when a tuple in the referenced table is updated. For example, if a customer's name changes in the "customers" table, all orders associated with that customer could be automatically updated to reflect the new name. This can be a convenient way to maintain data consistency, but it can also lead to unexpected changes if not used carefully.
3) Set null values: With this approach, the DBMS will automatically set the foreign key values in the referencing table to null when a tuple in the referenced table is updated or

deleted. This can be useful if you want to allow tuples to exist in the referencing table even if their referenced tuples are deleted, but it can also lead to data inconsistencies if not used carefully.

Therefore, we can use the query ON DELETE CASCADE and ON UPDATE CASCADE for the above mentioned cascade update to the references as shown below:

```
CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES library.author (author_id)ON DELETE CASCADE,
CONSTRAINT fk_publisher FOREIGN KEY (publisher_id) REFERENCES library.publisher (publisher_id)ON UPDATE CASCADE
```

We can also use the query below:

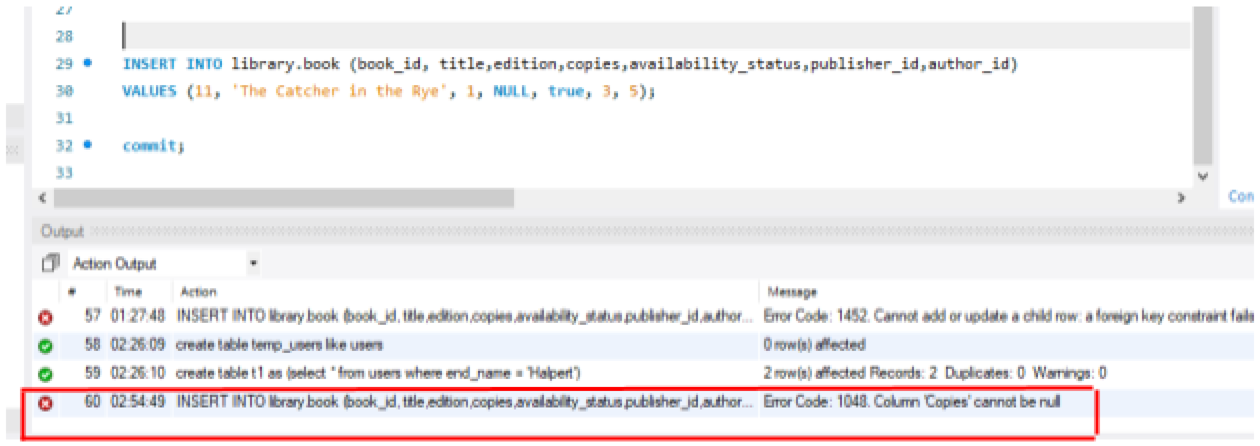
```
CONSTRAINT fk_author FOREIGN KEY (author_id) REFERENCES library.author (author_id)ON DELETE SET NULL,
CONSTRAINT fk_publisher FOREIGN KEY (publisher_id) REFERENCES library.publisher (publisher_id)ON UPDATE SET NULL
```

G1 and G2:

<u>3.3</u>

1) Attempt to insert book with null value of copies.

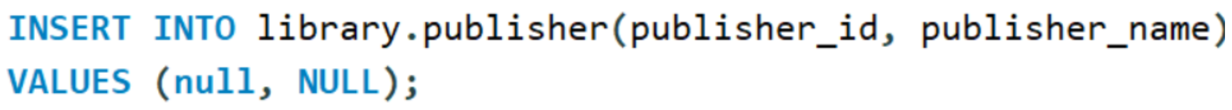
The below figure shows an error while executing the given query



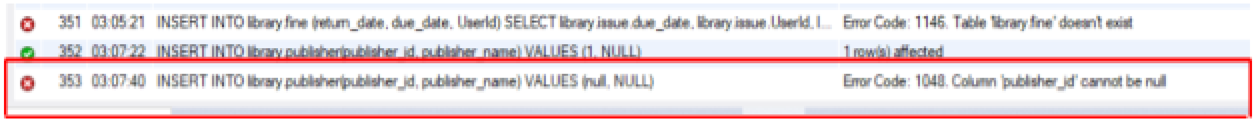
```
INSERT INTO library.book (book_id, title,edition,copies,availability_status,publisher_id,author_id)
VALUES (11, 'The Catcher in the Rye', 1, NULL, true, 3, 5);

commit;
```

```
Output
Action Output
 #   Time      Action                                                                       Message
 57  01:27:48  INSERT INTO library.book (book_id, title,edition,copies,availability_status,publisher_id,author...  Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails
 58  02:26:09  create table temp_users like users                                           0 row(s) affected
 59  02:26:10  create table t1 as (select * from users where end_name = 'Halpert')          2 row(s) affected Records: 2  Duplicates: 0  Warnings: 0
 60  02:54:49  INSERT INTO library.book (book_id, title,edition,copies,availability_status,publisher_id,author...  Error Code: 1048. Column 'Copies' cannot be null
```

2) This query attempts to insert a new row into the publisher without specifying a value for the id column, which has a NOT NULL constraint. This would result in a constraint violation error, as a publisher id is required for all rows in the table.

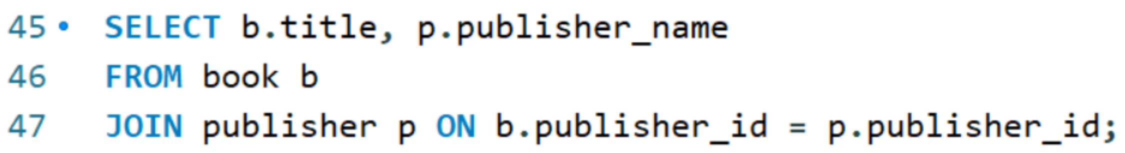


```sql
INSERT INTO library.publisher(publisher_id, publisher_name)
VALUES (null, NULL);
```

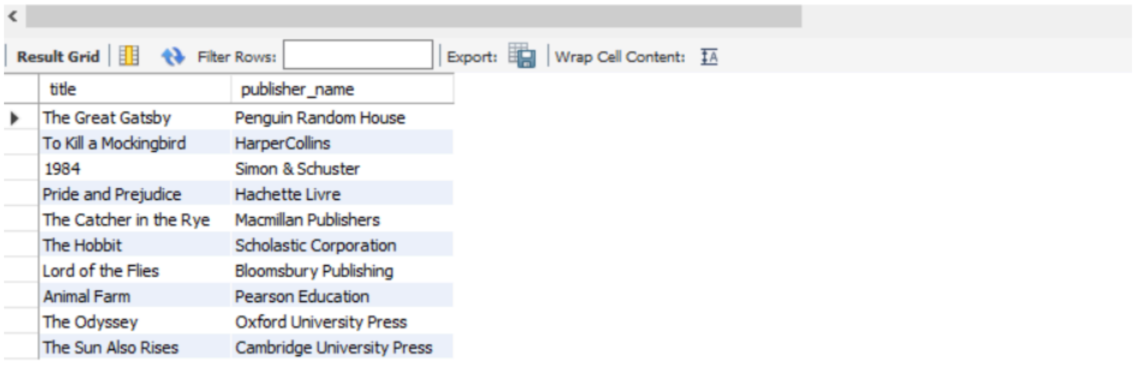


```
351  03:05:21  INSERT INTO library.fine (return_date, due_date, UserId) SELECT library.issue.due_date, library.issue.UserId, l...  Error Code: 1146. Table 'library.fine' doesn't exist
352  03:07:22  INSERT INTO library.publisher(publisher_id, publisher_name) VALUES (1, NULL)                                          1 row(s) affected
353  03:07:40  INSERT INTO library.publisher(publisher_id, publisher_name) VALUES (null, NULL)                                       Error Code: 1048. Column 'publisher_id' cannot be null
```

3) Retrieve the book titles and corresponding publisher names for all books in the library

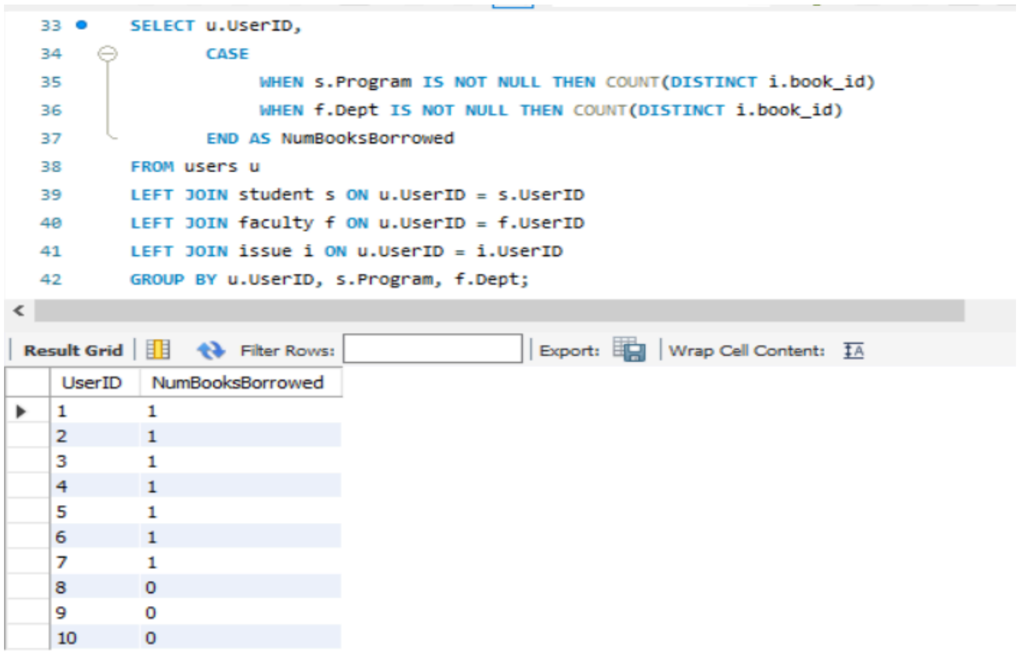


```sql
SELECT b.title, p.publisher_name
FROM book b
JOIN publisher p ON b.publisher_id = p.publisher_id;
```



title	publisher_name
The Great Gatsby	Penguin Random House
To Kill a Mockingbird	HarperCollins
1984	Simon & Schuster
Pride and Prejudice	Hachette Livre
The Catcher in the Rye	Macmillan Publishers
The Hobbit	Scholastic Corporation
Lord of the Flies	Bloomsbury Publishing
Animal Farm	Pearson Education
The Odyssey	Oxford University Press
The Sun Also Rises	Cambridge University Press

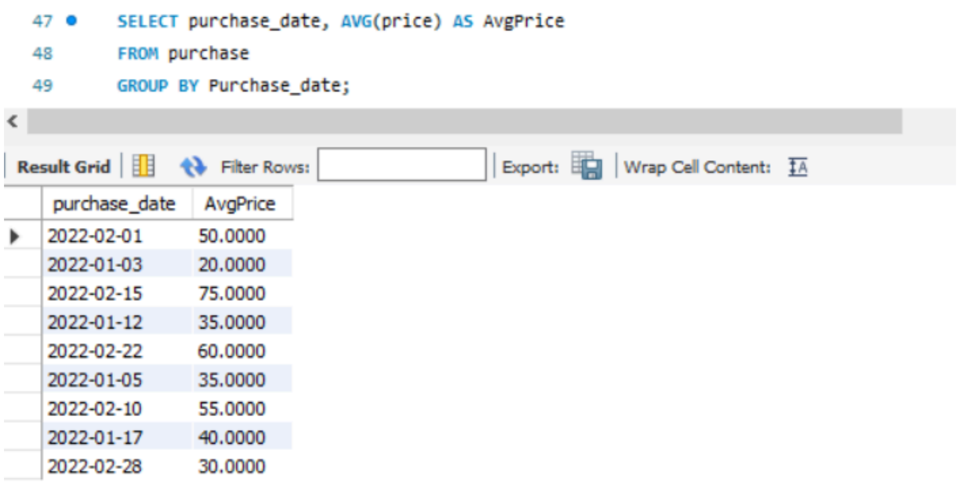
4) Retrieve the user IDs and their corresponding number of borrowed books for all students and faculty members:



```
SELECT u.UserID,
       CASE
           WHEN s.Program IS NOT NULL THEN COUNT(DISTINCT i.book_id)
           WHEN f.Dept IS NOT NULL THEN COUNT(DISTINCT i.book_id)
       END AS NumBooksBorrowed
FROM users u
LEFT JOIN student s ON u.UserID = s.UserID
LEFT JOIN faculty f ON u.UserID = f.UserID
LEFT JOIN issue i ON u.UserID = i.UserID
GROUP BY u.UserID, s.Program, f.Dept;
```

UserID	NumBooksBorrowed
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	0
9	0
10	0

5) Retrieve the average price of all books purchased on each day:



```
SELECT purchase_date, AVG(price) AS AvgPrice
FROM purchase
GROUP BY Purchase_date;
```

purchase_date	AvgPrice
2022-02-01	50.0000
2022-01-03	20.0000
2022-02-15	75.0000
2022-01-12	35.0000
2022-02-22	60.0000
2022-01-05	35.0000
2022-02-10	55.0000
2022-01-17	40.0000
2022-02-28	30.0000

Contribution:

Team- G1:

1) Sankarshan Kulkarni (20110184) - Schema relation constraints
2) Riya Dhantoliya (20110168) - Schema relation constriants
3) Gajanan Donge (20110061) - All work in 3.1 and 3.3
4) Jayesh Bhadange (20110082) - All work in 3.1 and 3.3

Team - G2: collectively did the G2 work

1) Pranav Rathod (20110143)
2) Naval Jaggi (20110118)
3) Harendra Khatik (20110072)
4) Sparsh Dawra (20110203)
5) Manpreet Singh (20110109)
6) Siddarth Joshi(19110169)