College Library Management System

Topic covers:

- Objective
- Key Features
- Benefits
- E-R Diagram
- Create and use Database
- Create Table
- Insert Value in table
- Create and run query
- Mind Map
- Test Case Writing
- Test Report
- Bug Reporting
- Test metrics
- Conclusion notes

Objective:

The College Library Management System is a database project designed to efficiently manage and organize the resources of a college library. This system aims to streamline the entire library process, from cataloging and tracking books to managing borrower information and facilitating seamless checkouts and returns. By leveraging a robust database, the system enhances the overall user experience for both librarians and students, promoting a more effective learning environment.

Key Features:

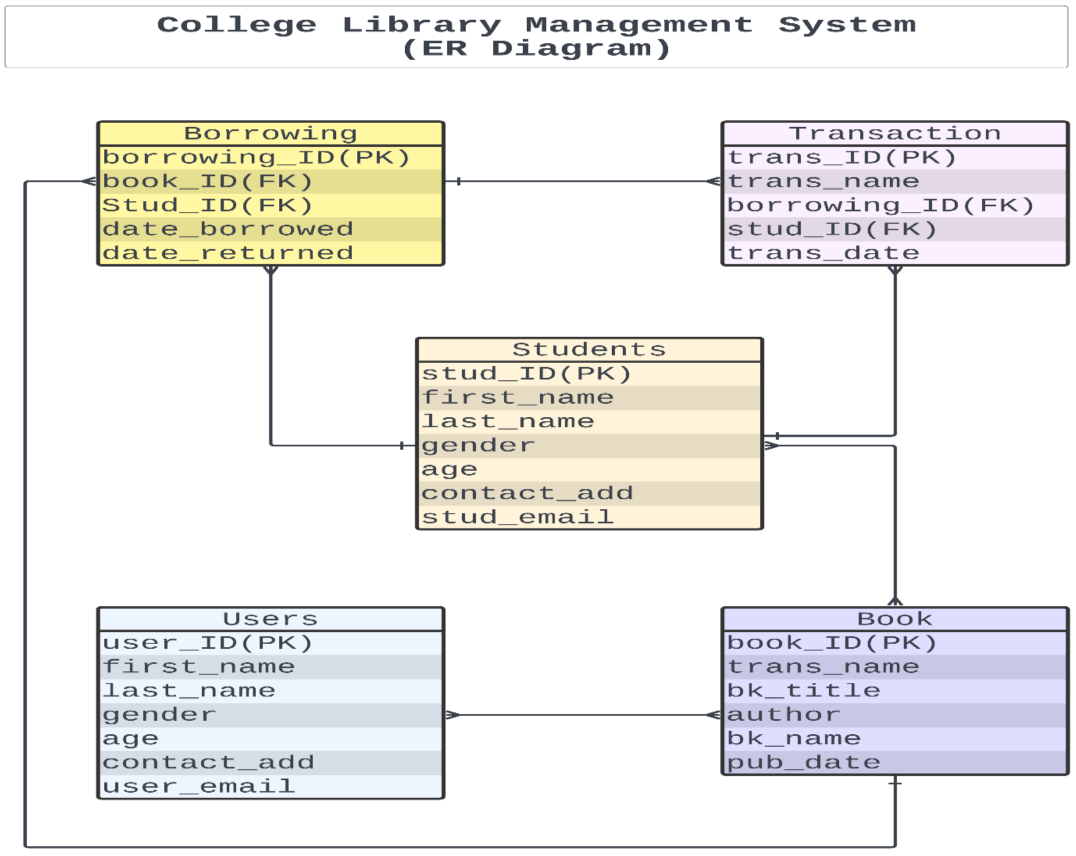
- Book Cataloging: The system allows librarians to input and update book details, including titles, authors, ISBNs, genres, and publication information. This feature enables easy search and retrieval of books from the database.
- Student Records: The database stores essential student information, such as names, student IDs, contact details, and course enrollments. This data is utilized to manage the borrowing and returning of books by students.
- Borrowing and Returning: Students can borrow books by providing their student IDs, and the system updates the database accordingly. Similarly, when returning books, the database is updated to reflect the availability of books.
- Book Availability and Tracking: The system maintains real-time information about the availability and location of each book. Students can check the availability of a book before visiting the library.
- Notifications and Reminders: The system sends automated notifications to remind students of upcoming due dates or any overdue books. Additionally, it informs librarians about any books that are due for return.

- Fine Management: The system calculates and manages fines for late book returns automatically. It also keeps track of fine payments made by students.
- Reporting and Analytics: The system generates comprehensive reports on various aspects, such as most borrowed books, popular genres, late returns, and fine collections. These reports aid librarians in making data-driven decisions.
- Security and User Access: The system ensures secure access, allowing different levels of permissions for administrators, librarians, and students.

Benefits:

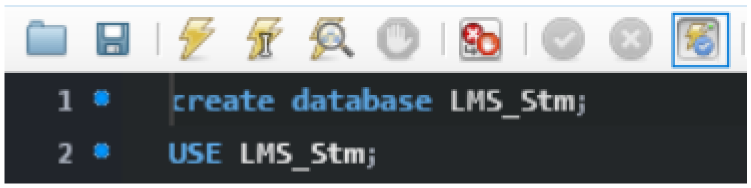
- Enhanced Efficiency: The College Library Management System streamlines library operations, reducing manual tasks, and improving overall efficiency.
- Improved Resource Management: By providing real-time information on book availability and usage patterns, the system optimizes resource allocation and procurement decisions.
- Better User Experience: Students can easily search for and check out books, while librarians can efficiently manage the library's collection.
- Time and Cost Savings: Automation of various tasks reduces the time and effort spent on administrative work, leading to cost savings for the college.
- Data-Driven Decisions: The system's analytics and reports enable administrators to make informed decisions regarding library policies and resource allocation.

E-R Diagram:



Create and use Database:

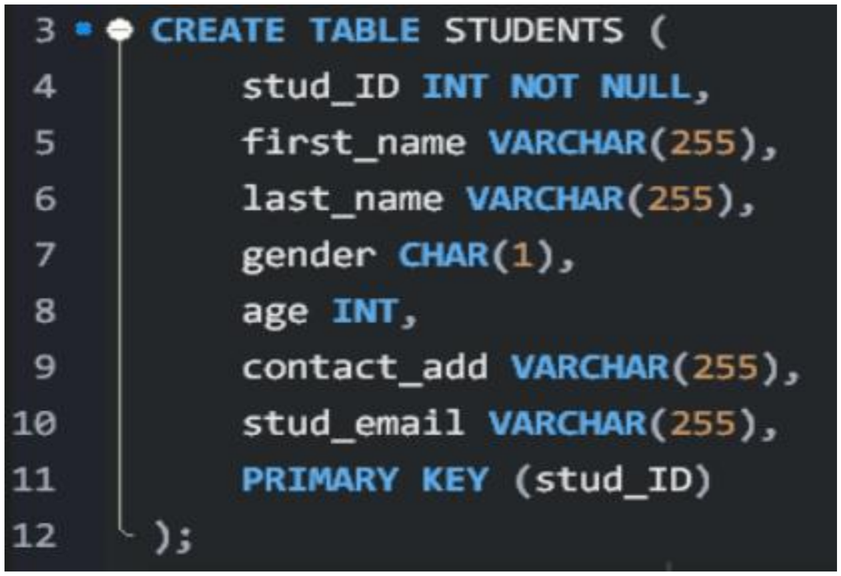
- At first, I **create** Database as **LMS_Stm;**
- Then I use this database.



```
create database LMS_Stm;
USE LMS_Stm;
```

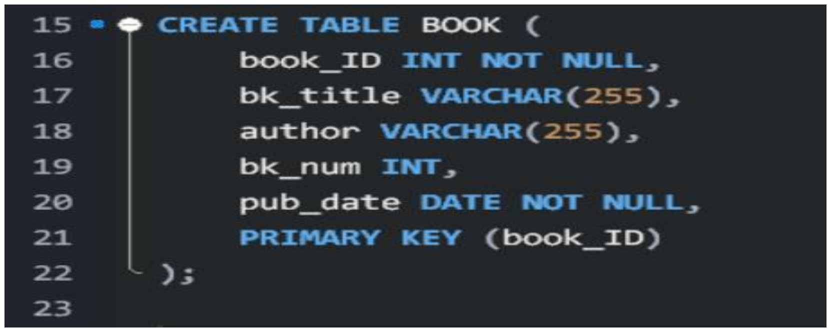
Create Table:

- **Create "STUDENTS" Table:** I create **"STUDENTS"** table.



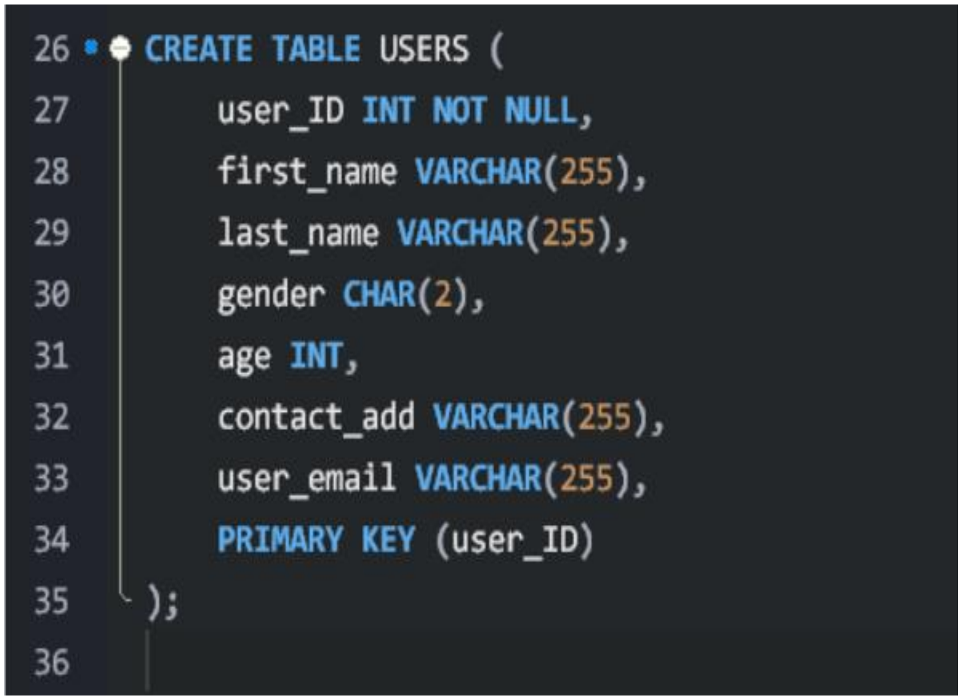
```
CREATE TABLE STUDENTS (
        stud_ID INT NOT NULL,
        first_name VARCHAR(255),
        last_name VARCHAR(255),
        gender CHAR(1),
        age INT,
        contact_add VARCHAR(255),
        stud_email VARCHAR(255),
        PRIMARY KEY (stud_ID)
);
```

- **Create "Book" Table:** I create **"Book"** table.



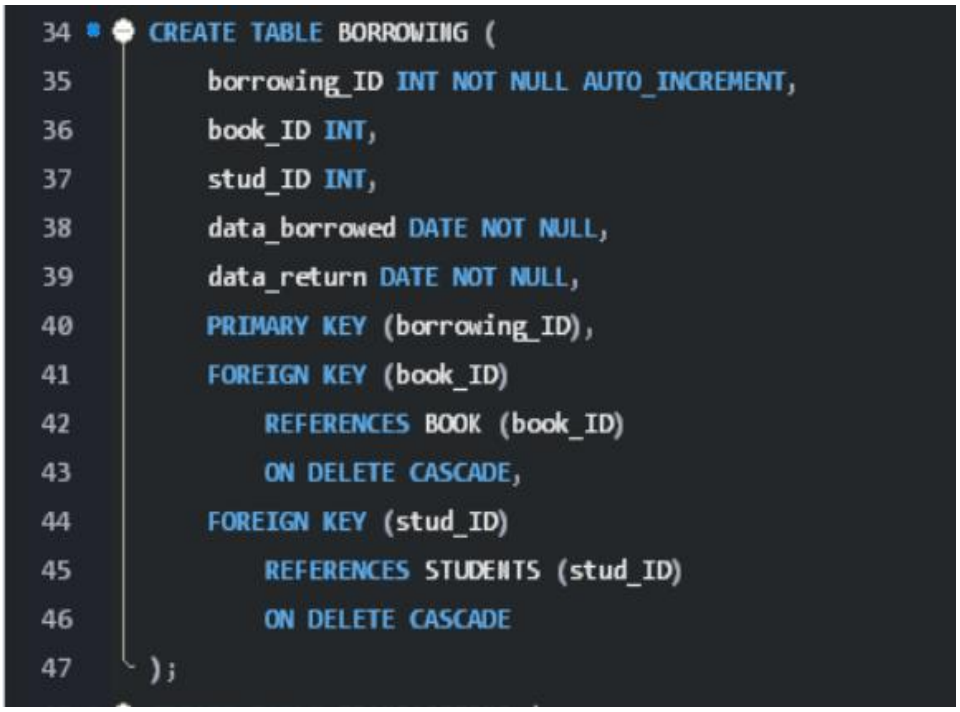
```
CREATE TABLE BOOK (
        book_ID INT NOT NULL,
        bk_title VARCHAR(255),
        author VARCHAR(255),
        bk_num INT,
        pub_date DATE NOT NULL,
        PRIMARY KEY (book_ID)
);
```

- **Create “USERS” Table:** I create **“USERS”** table.



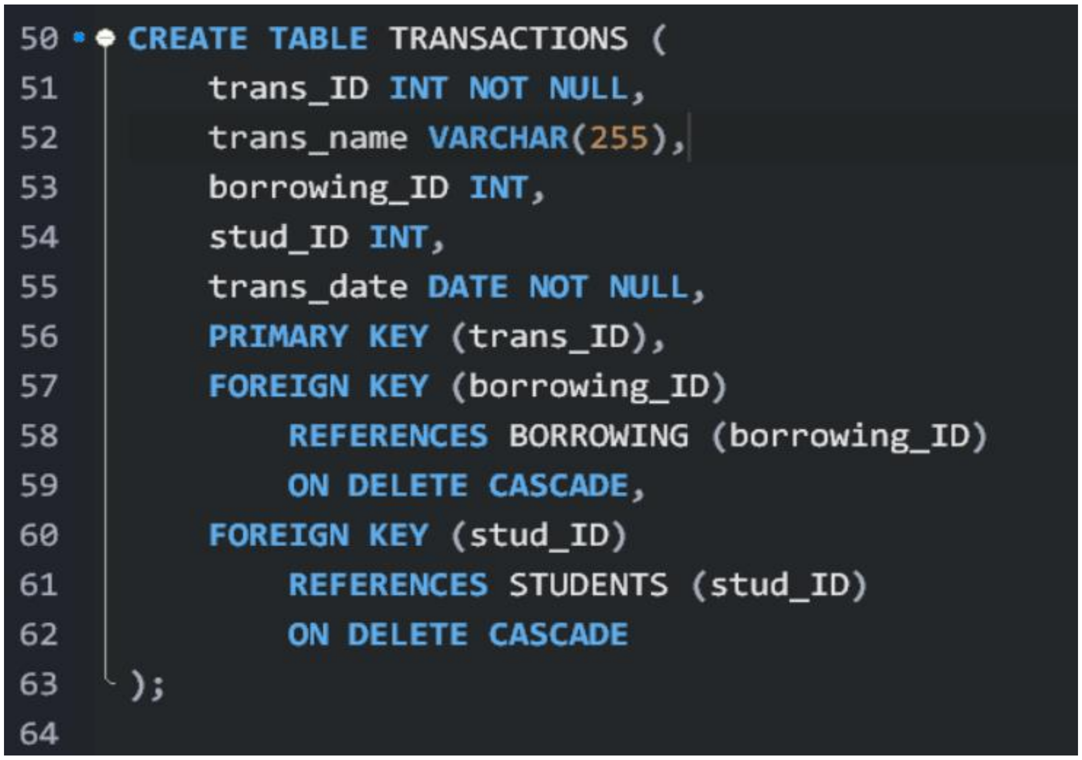
```sql
CREATE TABLE USERS (
    user_ID INT NOT NULL,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    gender CHAR(2),
    age INT,
    contact_add VARCHAR(255),
    user_email VARCHAR(255),
    PRIMARY KEY (user_ID)
);
```

- **Create “BORROWING” Table: “BORROWING”** table has been created.



```sql
CREATE TABLE BORROWING (
    borrowing_ID INT NOT NULL AUTO_INCREMENT,
    book_ID INT,
    stud_ID INT,
    data_borrowed DATE NOT NULL,
    data_return DATE NOT NULL,
    PRIMARY KEY (borrowing_ID),
    FOREIGN KEY (book_ID)
        REFERENCES BOOK (book_ID)
        ON DELETE CASCADE,
    FOREIGN KEY (stud_ID)
        REFERENCES STUDENTS (stud_ID)
        ON DELETE CASCADE
);
```

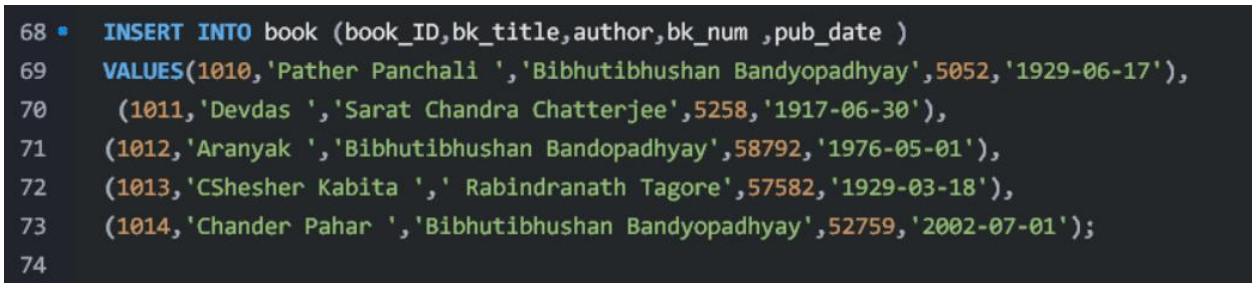
- **Create “TRANSACTIONS” Table: “TRANSACTIONS”** table has been created.



```sql
CREATE TABLE TRANSACTIONS (
    trans_ID INT NOT NULL,
    trans_name VARCHAR(255),
    borrowing_ID INT,
    stud_ID INT,
    trans_date DATE NOT NULL,
    PRIMARY KEY (trans_ID),
    FOREIGN KEY (borrowing_ID)
        REFERENCES BORROWING (borrowing_ID)
        ON DELETE CASCADE,
    FOREIGN KEY (stud_ID)
        REFERENCES STUDENTS (stud_ID)
        ON DELETE CASCADE
);
```

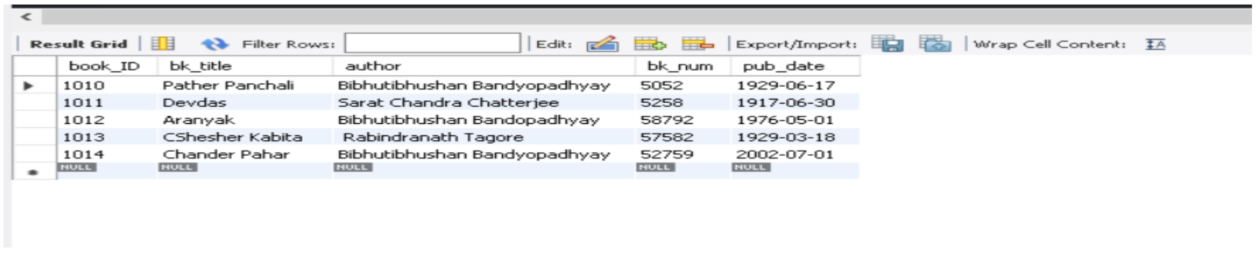
Insert Value in table:

- **Insert** Value in **”book”** table**: ”book” table value has been inserted.**



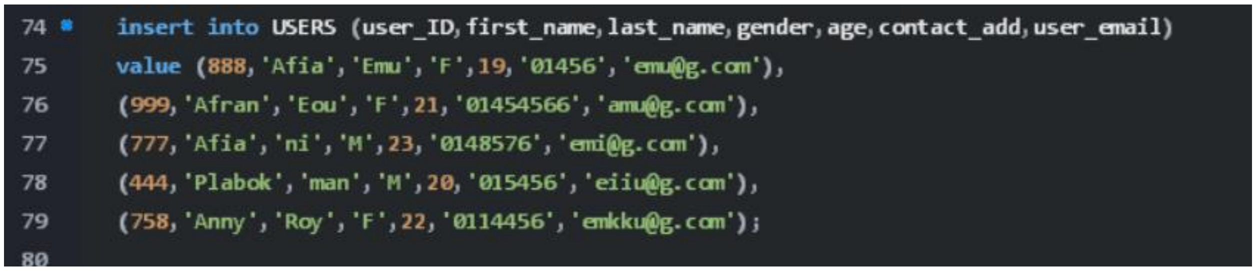
```sql
INSERT INTO book (book_ID,bk_title,author,bk_num ,pub_date )
VALUES(1010,'Pather Panchali ','Bibhutibhushan Bandyopadhyay',5052,'1929-06-17'),
 (1011,'Devdas ','Sarat Chandra Chatterjee',5258,'1917-06-30'),
(1012,'Aranyak ','Bibhutibhushan Bandopadhyay',58792,'1976-05-01'),
(1013,'CShesher Kabita ',' Rabindranath Tagore',57582,'1929-03-18'),
(1014,'Chander Pahar ','Bibhutibhushan Bandyopadhyay',52759,'2002-07-01');
```

- **Here we can see this output by this query: select * from book;**



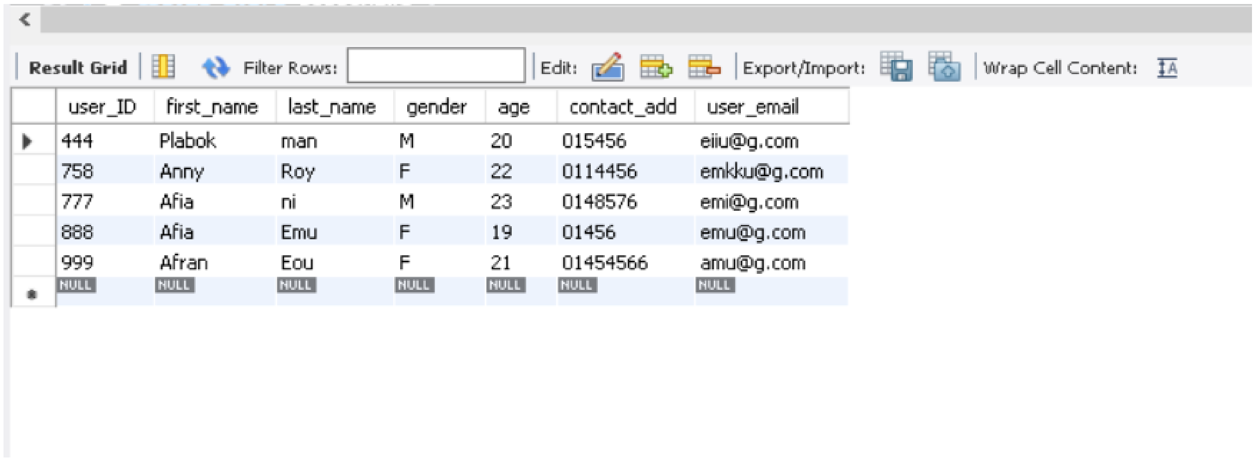
book_ID	bk_title	author	bk_num	pub_date
1010	Pather Panchali	Bibhutibhushan Bandyopadhyay	5052	1929-06-17
1011	Devdas	Sarat Chandra Chatterjee	5258	1917-06-30
1012	Aranyak	Bibhutibhushan Bandopadhyay	58792	1976-05-01
1013	CShesher Kabita	Rabindranath Tagore	57582	1929-03-18
1014	Chander Pahar	Bibhutibhushan Bandyopadhyay	52759	2002-07-01
NULL	NULL	NULL	NULL	NULL

- **Insert** Value in **"User"** table**: "User" table value has been inserted.**



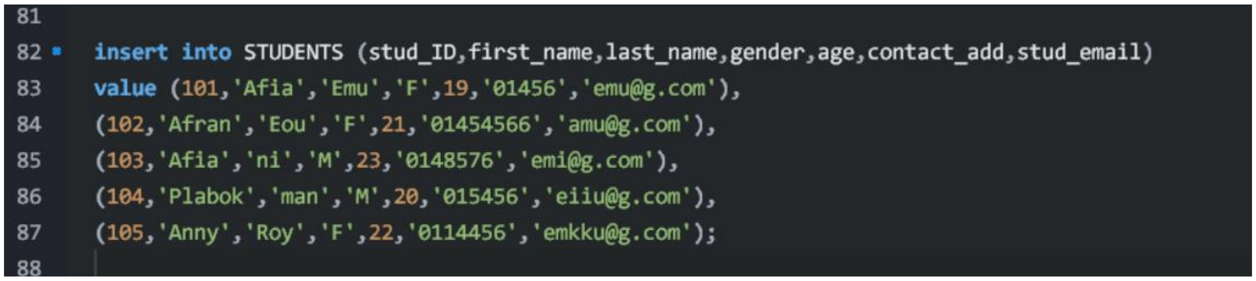
```
insert into USERS (user_ID, first_name, last_name, gender, age, contact_add, user_email)
value (888,'Afia','Emu','F',19,'01456','emu@g.com'),
(999,'Afran','Eou','F',21,'01454566','amu@g.com'),
(777,'Afia','ni','M',23,'0148576','emi@g.com'),
(444,'Plabok','man','M',20,'015456','eiiu@g.com'),
(758,'Anny','Roy','F',22,'0114456','emkku@g.com');
```

- **Here we can see this output by this query: select * from USERS;**



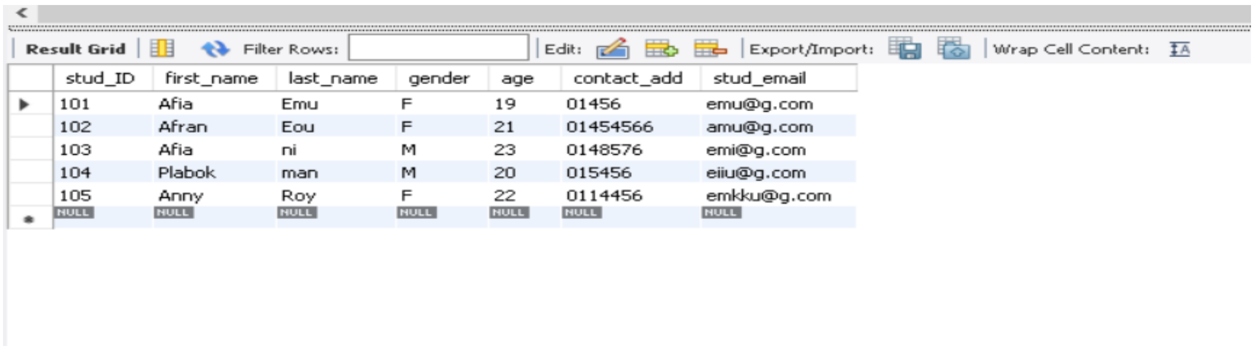
user_ID	first_name	last_name	gender	age	contact_add	user_email
444	Plabok	man	M	20	015456	eiiu@g.com
758	Anny	Roy	F	22	0114456	emkku@g.com
777	Afia	ni	M	23	0148576	emi@g.com
888	Afia	Emu	F	19	01456	emu@g.com
999	Afran	Eou	F	21	01454566	amu@g.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **Insert** Value in **"Student"** table**: "Student" table value has been inserted.**



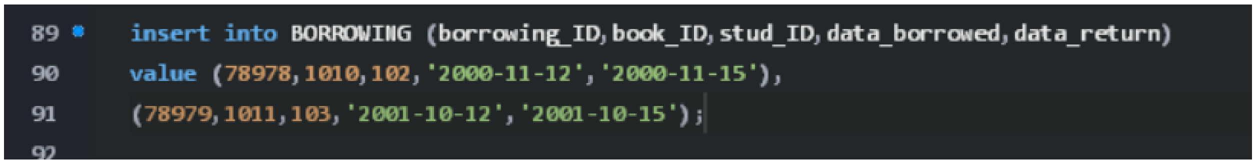
```
insert into STUDENTS (stud_ID,first_name,last_name,gender,age,contact_add,stud_email)
value (101,'Afia','Emu','F',19,'01456','emu@g.com'),
(102,'Afran','Eou','F',21,'01454566','amu@g.com'),
(103,'Afia','ni','M',23,'0148576','emi@g.com'),
(104,'Plabok','man','M',20,'015456','eiiu@g.com'),
(105,'Anny','Roy','F',22,'0114456','emkku@g.com');
```

- **Here we can see this output by this query: select * from STUDENTS;**



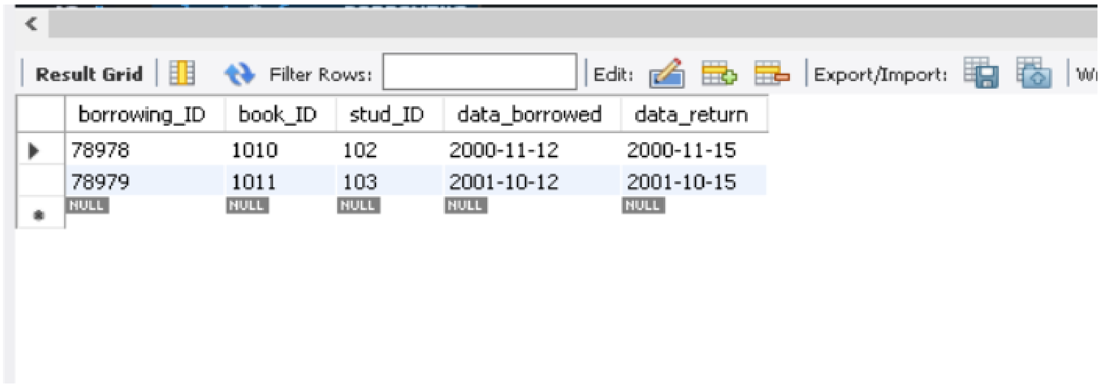
stud_ID	first_name	last_name	gender	age	contact_add	stud_email
101	Afia	Emu	F	19	01456	emu@g.com
102	Afran	Eou	F	21	01454566	amu@g.com
103	Afia	ni	M	23	0148576	emi@g.com
104	Plabok	man	M	20	015456	eiiu@g.com
105	Anny	Roy	F	22	0114456	emkku@g.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **Insert** Value in **"Borrowing"** table**: "Borrowing" table value has been inserted.**



```
insert into BORROWING (borrowing_ID, book_ID, stud_ID, data_borrowed, data_return)
value (78978,1010,102,'2000-11-12','2000-11-15'),
(78979,1011,103,'2001-10-12','2001-10-15');
```

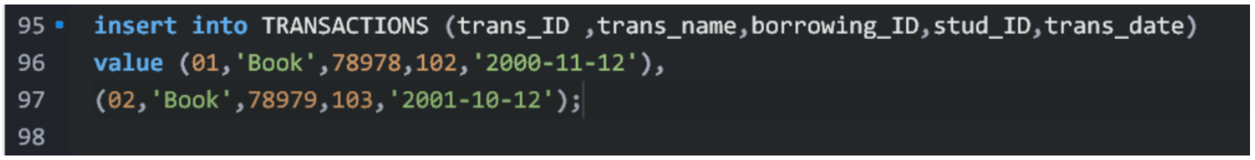
- **Here we can see this output by this query: select * from STUDENTS;**



Result Grid | Filter Rows: | Edit: | Export/Import: | Wi

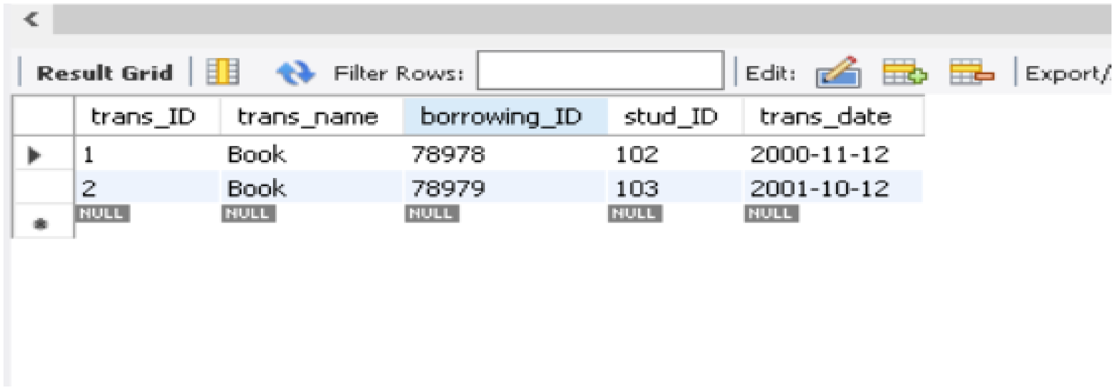
borrowing_ID	book_ID	stud_ID	data_borrowed	data_return
78978	1010	102	2000-11-12	2000-11-15
78979	1011	103	2001-10-12	2001-10-15
NULL	NULL	NULL	NULL	NULL

- **Insert** Value in **" TRANSACTIONS"** table**: " TRANSACTIONS" table value has been inserted.**



```
insert into TRANSACTIONS (trans_ID ,trans_name,borrowing_ID,stud_ID,trans_date)
value (01,'Book',78978,102,'2000-11-12'),
(02,'Book',78979,103,'2001-10-12');
```

- **Here we can see this output by this query: select * from TRANSACTIONS;**

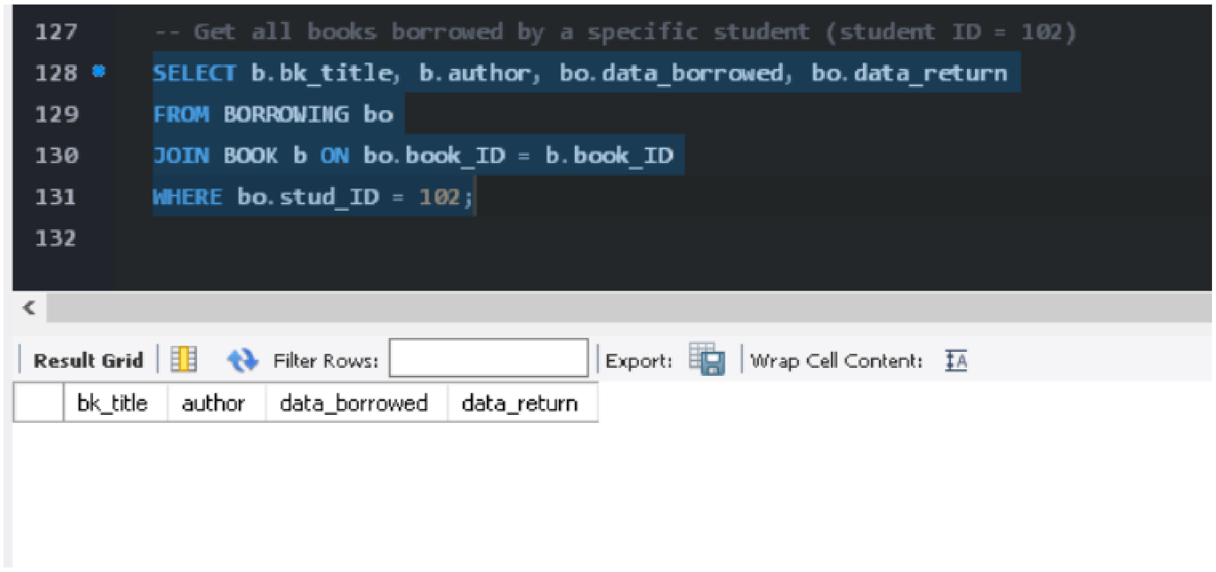


Result Grid | Filter Rows: | Edit: | Export/

trans_ID	trans_name	borrowing_ID	stud_ID	trans_date
1	Book	78978	102	2000-11-12
2	Book	78979	103	2001-10-12
NULL	NULL	NULL	NULL	NULL

Create and run query:

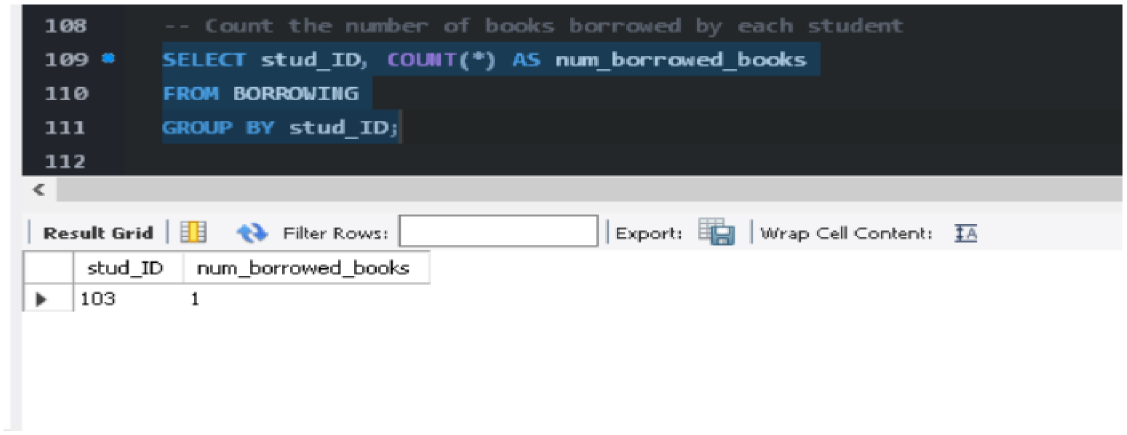
- **Get all book borrowed by a specific student (Student ID=102)**



```
-- Get all books borrowed by a specific student (student ID = 102)
SELECT b.bk_title, b.author, bo.data_borrowed, bo.data_return
FROM BORROWING bo
JOIN BOOK b ON bo.book_ID = b.book_ID
WHERE bo.stud_ID = 102;
```

bk_title	author	data_borrowed	data_return

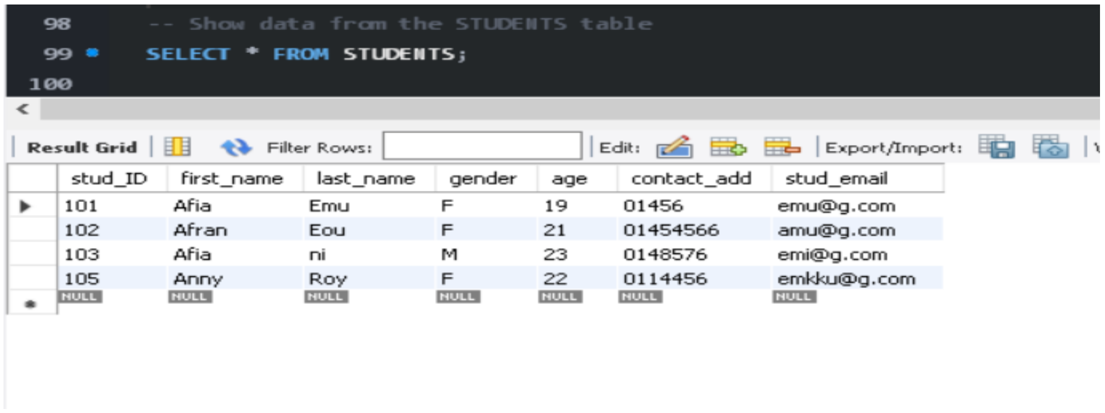
- **Count the number of books borrowed by each student.**



```
-- Count the number of books borrowed by each student
SELECT stud_ID, COUNT(*) AS num_borrowed_books
FROM BORROWING
GROUP BY stud_ID;
```

stud_ID	num_borrowed_books
103	1

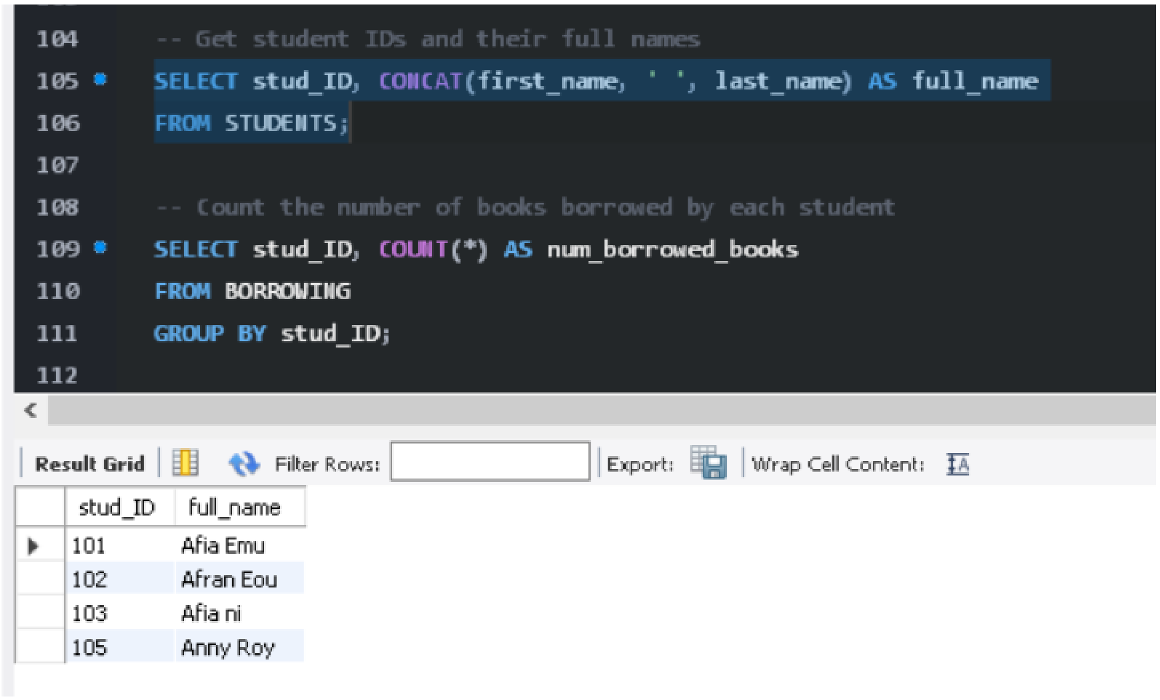
- Show data from the STUDENTS table.



```
-- Show data from the STUDENTS table
SELECT * FROM STUDENTS;
```

stud_ID	first_name	last_name	gender	age	contact_add	stud_email
101	Afia	Emu	F	19	01456	emu@g.com
102	Afran	Eou	F	21	01454566	amu@g.com
103	Afia	ni	M	23	0148576	emi@g.com
105	Anny	Roy	F	22	0114456	emkku@g.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **Get student id and their full names.**



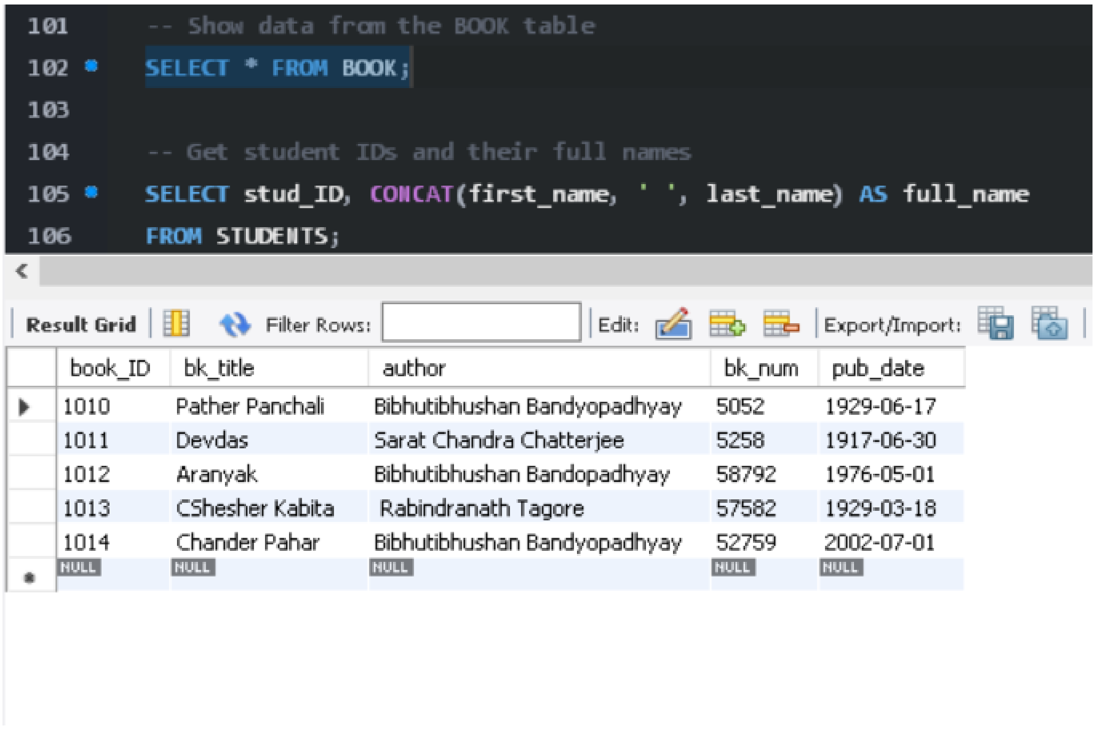
```
-- Get student IDs and their full names
SELECT stud_ID, CONCAT(first_name, ' ', last_name) AS full_name
FROM STUDENTS;

-- Count the number of books borrowed by each student
SELECT stud_ID, COUNT(*) AS num_borrowed_books
FROM BORROWING
GROUP BY stud_ID;
```

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

stud_ID	full_name
101	Afia Emu
102	Afran Eou
103	Afia ni
105	Anny Roy

- **Show data from book table.**



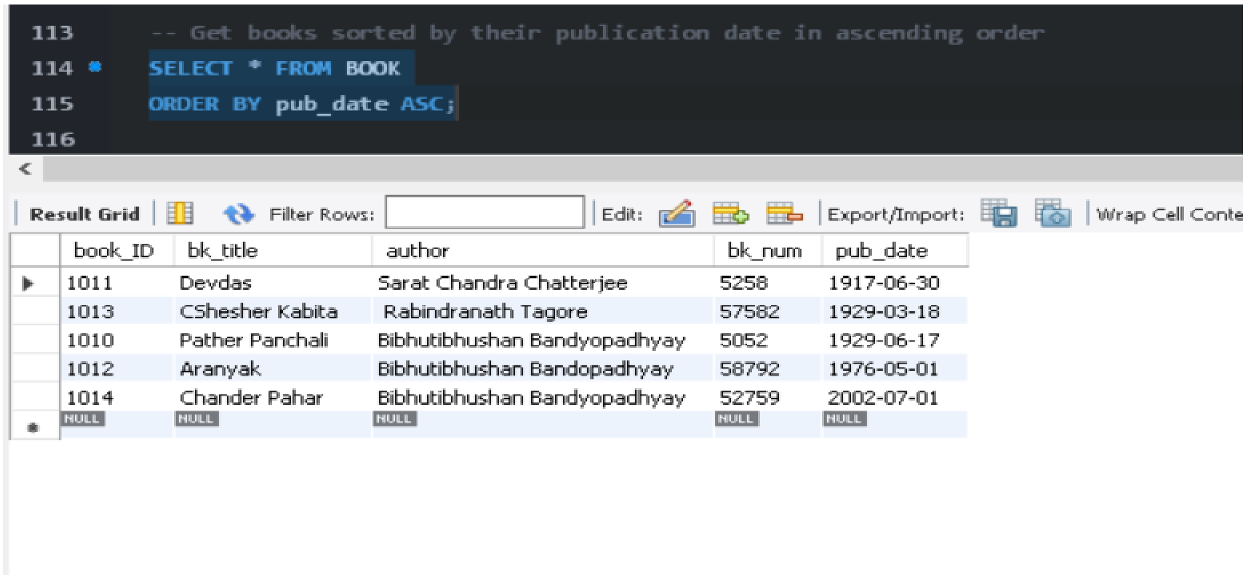
```
-- Show data from the BOOK table
SELECT * FROM BOOK;

-- Get student IDs and their full names
SELECT stud_ID, CONCAT(first_name, ' ', last_name) AS full_name
FROM STUDENTS;
```

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

book_ID	bk_title	author	bk_num	pub_date
1010	Pather Panchali	Bibhutibhushan Bandyopadhyay	5052	1929-06-17
1011	Devdas	Sarat Chandra Chatterjee	5258	1917-06-30
1012	Aranyak	Bibhutibhushan Bandopadhyay	58792	1976-05-01
1013	CShesher Kabita	Rabindranath Tagore	57582	1929-03-18
1014	Chander Pahar	Bibhutibhushan Bandyopadhyay	52759	2002-07-01
NULL	NULL	NULL	NULL	NULL

- **Get book sorted by their publication date in ascending order.**

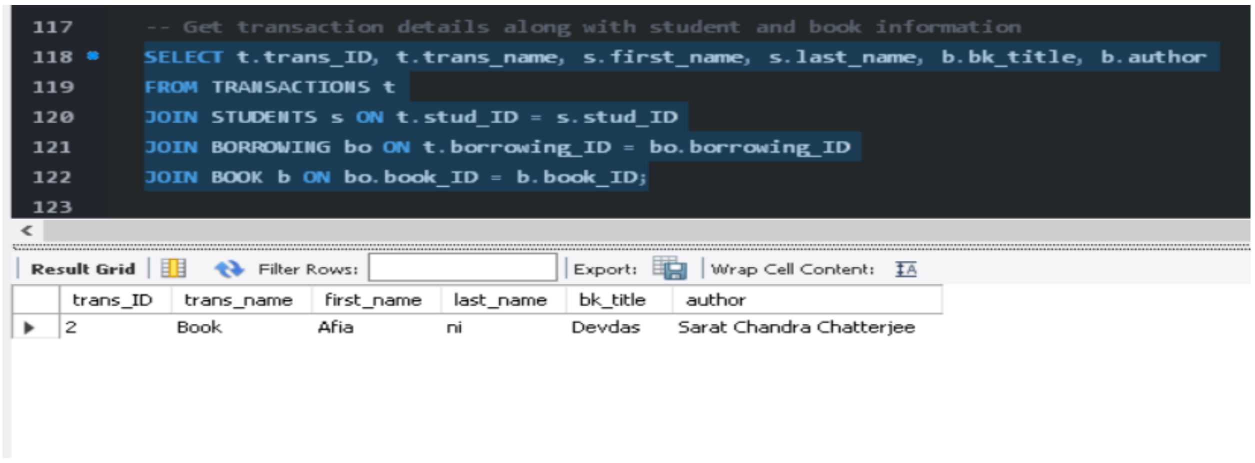


```
-- Get books sorted by their publication date in ascending order
SELECT * FROM BOOK
ORDER BY pub_date ASC;
```

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

book_ID	bk_title	author	bk_num	pub_date
1011	Devdas	Sarat Chandra Chatterjee	5258	1917-06-30
1013	CShesher Kabita	Rabindranath Tagore	57582	1929-03-18
1010	Pather Panchali	Bibhutibhushan Bandyopadhyay	5052	1929-06-17
1012	Aranyak	Bibhutibhushan Bandopadhyay	58792	1976-05-01
1014	Chander Pahar	Bibhutibhushan Bandyopadhyay	52759	2002-07-01
NULL	NULL	NULL	NULL	NULL

- **Get transaction details along with student and book information.**

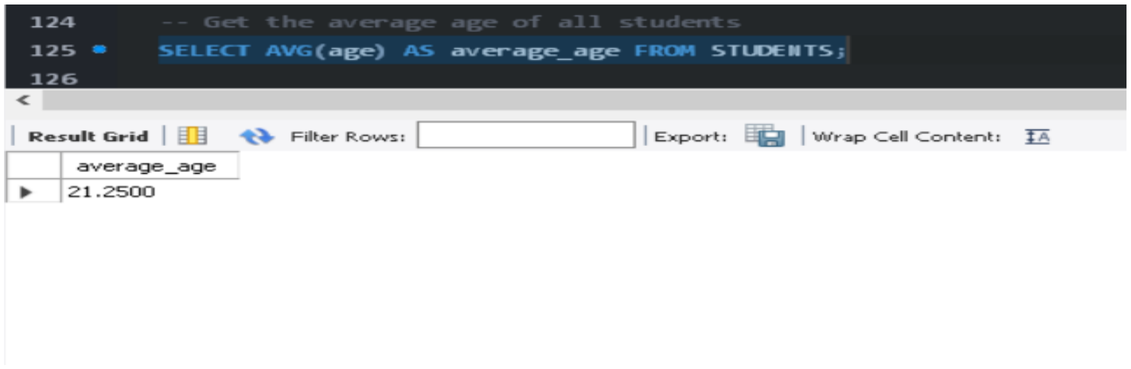


```
-- Get transaction details along with student and book information
SELECT t.trans_ID, t.trans_name, s.first_name, s.last_name, b.bk_title, b.author
FROM TRANSACTIONS t
JOIN STUDENTS s ON t.stud_ID = s.stud_ID
JOIN BORROWING bo ON t.borrowing_ID = bo.borrowing_ID
JOIN BOOK b ON bo.book_ID = b.book_ID;
```

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

trans_ID	trans_name	first_name	last_name	bk_title	author
2	Book	Afia	ni	Devdas	Sarat Chandra Chatterjee

- **Get average age of all students.**

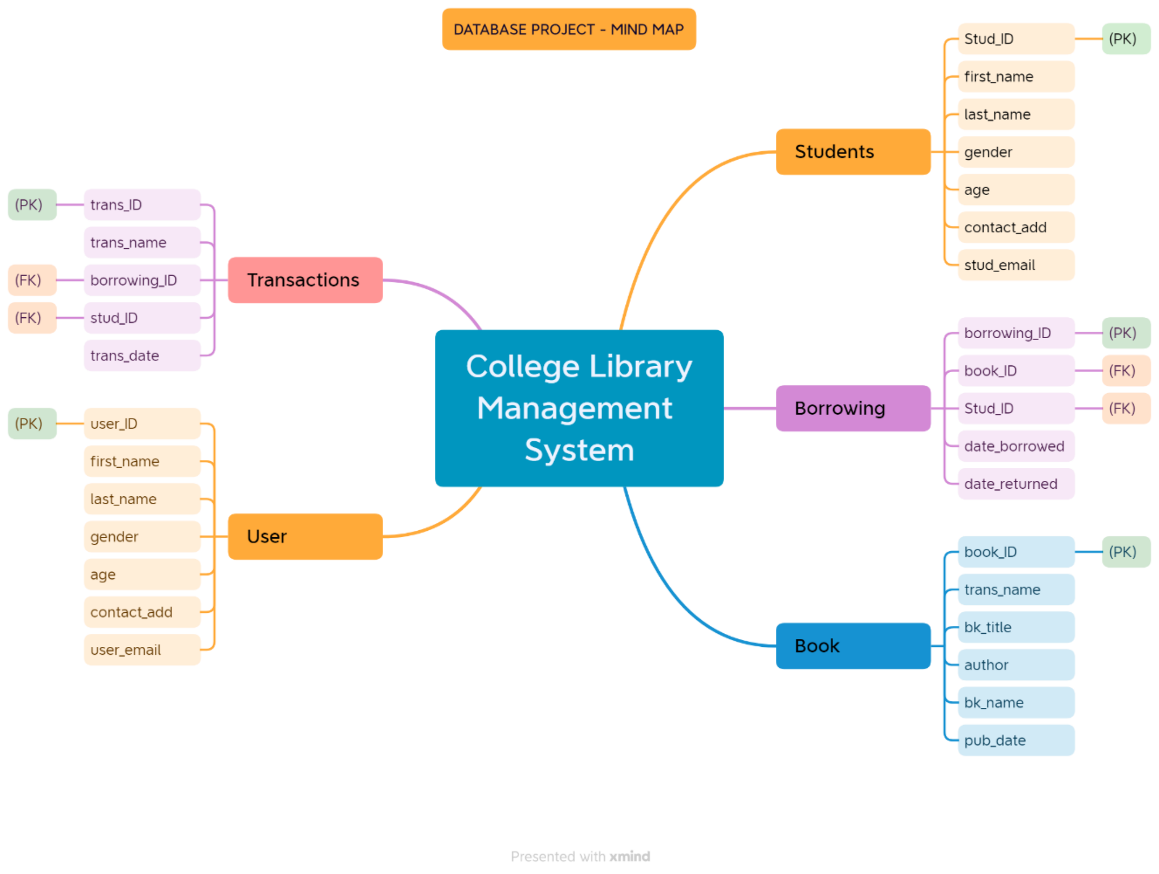


```
-- Get the average age of all students
SELECT AVG(age) AS average_age FROM STUDENTS;
```

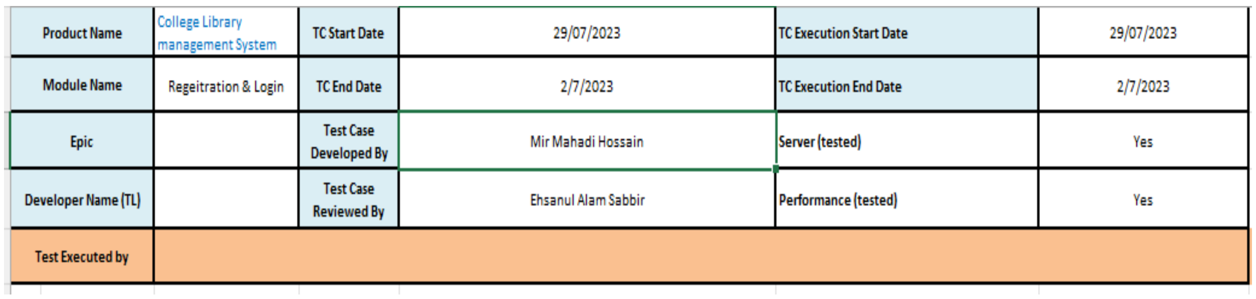
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

average_age
21.2500

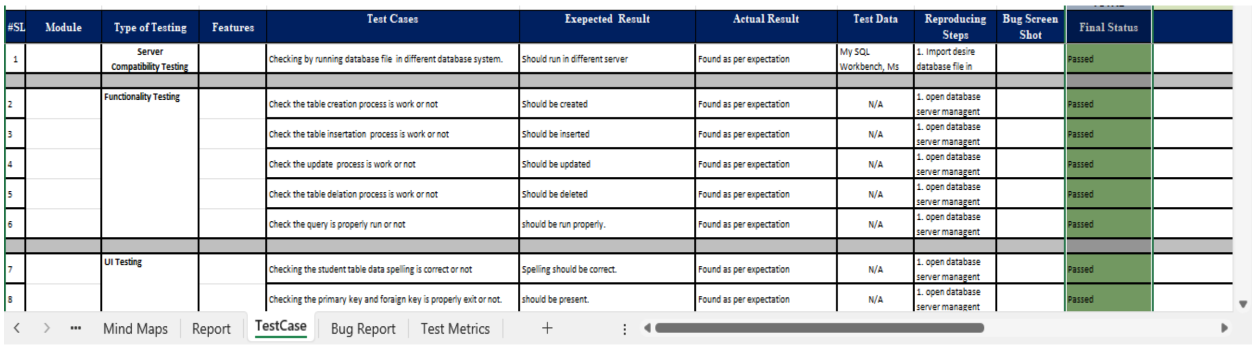
Mind Map of College Library management system:



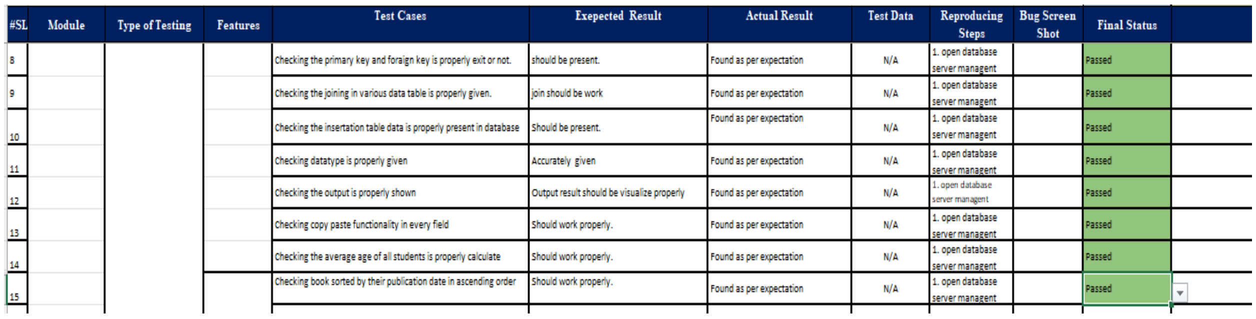
Test case writing:



Product Name	College Library management System	TC Start Date	29/07/2023	TC Execution Start Date	29/07/2023
Module Name	Regeitration & Login	TC End Date	2/7/2023	TC Execution End Date	2/7/2023
Epic		Test Case Developed By	Mir Mahadi Hossain	Server (tested)	Yes
Developer Name (TL)		Test Case Reviewed By	Ehsanul Alam Sabbir	Performance (tested)	Yes
Test Executed by					



#SL	Module	Type of Testing	Features	Test Cases	Exepected Result	Actual Result	Test Data	Reproducing Steps	Bug Screen Shot	Final Status	
1		Server Compatibility Testing		Checking by running database file in different database system.	Should run in different server	Found as per expectation	My SQL Workbench, Ms	1. Import desire database file in		Passed	
2		Functionality Testing		Check the table creation process is work or not	Should be created	Found as per expectation	N/A	1. open database server managent		Passed	
3				Check the table insertation process is work or not	Should be inserted	Found as per expectation	N/A	1. open database server managent		Passed	
4				Check the update process is work or not	Should be updated	Found as per expectation	N/A	1. open database server managent		Passed	
5				Check the table delation process is work or not	Should be deleted	Found as per expectation	N/A	1. open database server managent		Passed	
6				Check the query is properly run or not	should be run properly.	Found as per expectation	N/A	1. open database server managent		Passed	
7		UI Testing		Checking the student table data spelling is correct or not	Spelling should be correct.	Found as per expectation	N/A	1. open database server managent		Passed	
8				Checking the primary key and foraign key is properly exit or not.	should be present.	Found as per expectation	N/A	1. open database server managent		Passed	

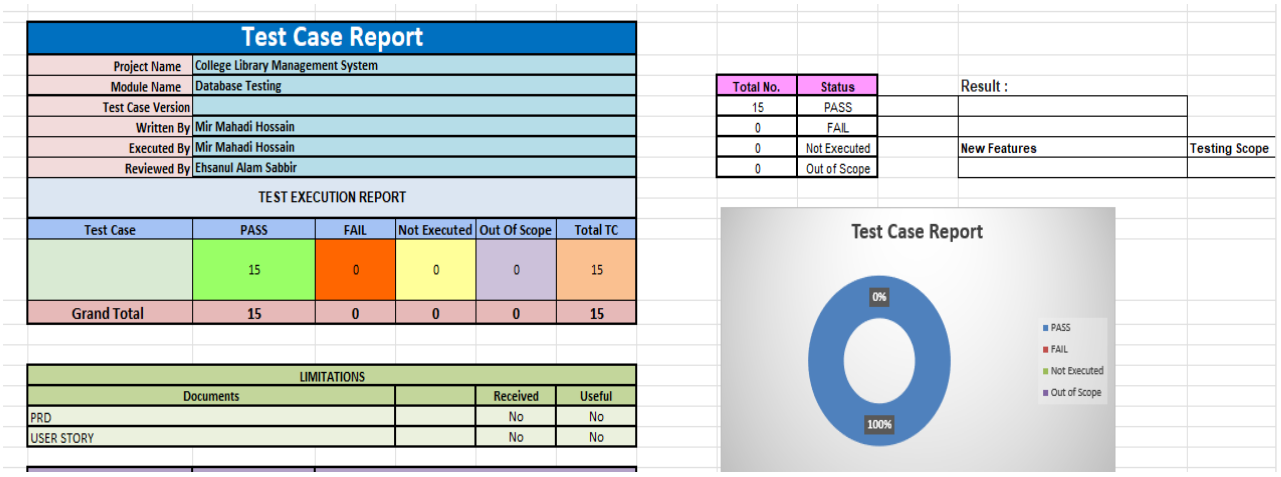


#SL	Module	Type of Testing	Features	Test Cases	Exepected Result	Actual Result	Test Data	Reproducing Steps	Bug Screen Shot	Final Status
8				Checking the primary key and foraign key is properly exit or not.	should be present.	Found as per expectation	N/A	1. open database server managent		Passed
9				Checking the joining in various data table is properly given.	join should be work	Found as per expectation	N/A	1. open database server managent		Passed
10				Checking the insertation table data is properly present in database	Should be present.	Found as per expectation	N/A	1. open database server managent		Passed
11				Checking datatype is properly given	Accurately given	Found as per expectation	N/A	1. open database server managent		Passed
12				Checking the output is properly shown	Output result should be visualize properly	Found as per expectation	N/A	1. open database server managent		Passed
13				Checking copy paste functionality in every field	Should work properly.	Found as per expectation	N/A	1. open database server managent		Passed
14				Checking the average age of all students is properly calculate	Should work properly.	Found as per expectation	N/A	1. open database server managent		Passed
15				Checking book sorted by their publication date in ascending order	Should work properly.	Found as per expectation	N/A	1. open database server managent		Passed

Test case summery

	TEST CASE
PASS	15
FAIL	0
Not Executed	0
Out of Scope	0
TOTAL	15

Total Test Report:



Test Case Report

Project Name	College Library Management System
Module Name	Database Testing
Test Case Version	
Written By	Mir Mahadi Hossain
Executed By	Mir Mahadi Hossain
Reviewed By	Ehsanul Alam Sabbir

TEST EXECUTION REPORT

Test Case	PASS	FAIL	Not Executed	Out Of Scope	Total TC
	15	0	0	0	15
Grand Total	15	0	0	0	15

LIMITATIONS

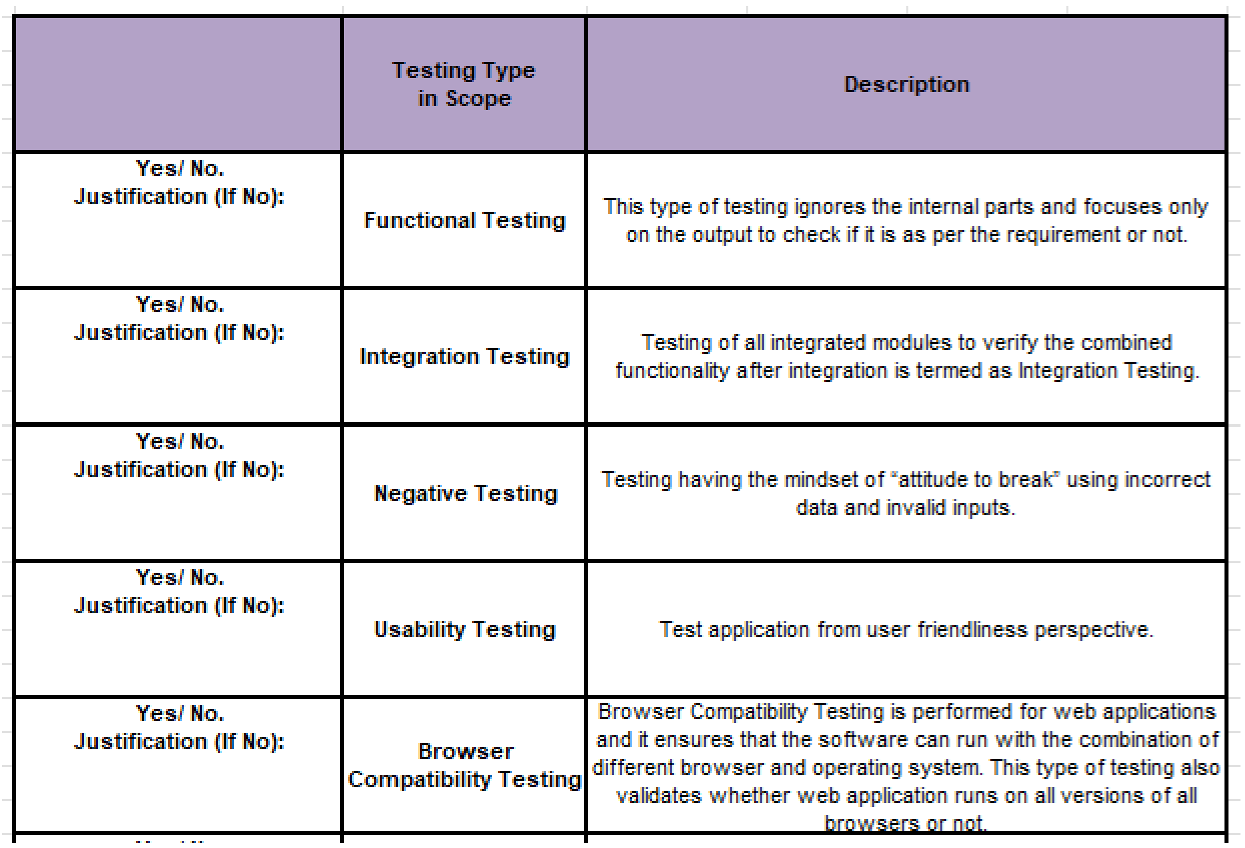
Documents		Received	Useful
PRD		No	No
USER STORY		No	No

Total No.	Status
15	PASS
0	FAIL
0	Not Executed
0	Out of Scope

Result :

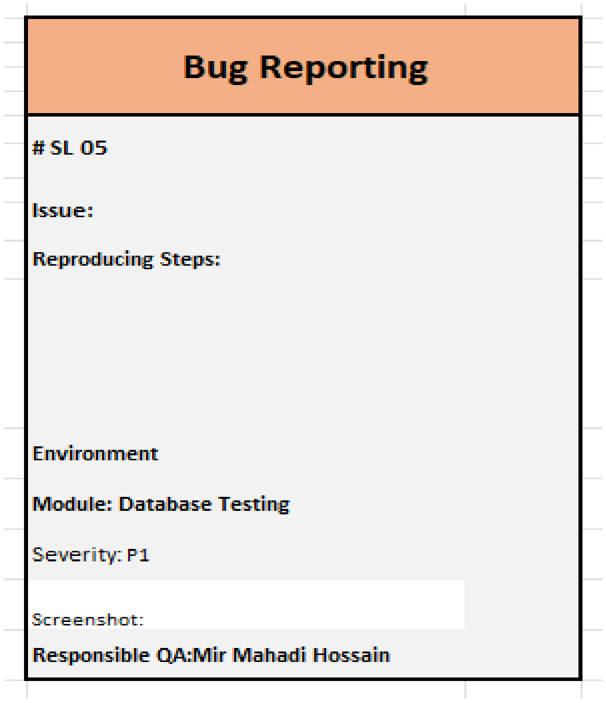
New Features	Testing Scope

Testing type in scope and description:



	Testing Type in Scope	Description
Yes/ No. Justification (If No):	Functional Testing	This type of testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not.
Yes/ No. Justification (If No):	Integration Testing	Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing.
Yes/ No. Justification (If No):	Negative Testing	Testing having the mindset of "attitude to break" using incorrect data and invalid inputs.
Yes/ No. Justification (If No):	Usability Testing	Test application from user friendliness perspective.
Yes/ No. Justification (If No):	Browser Compatibility Testing	Browser Compatibility Testing is performed for web applications and it ensures that the software can run with the combination of different browser and operating system. This type of testing also validates whether web application runs on all versions of all browsers or not.

Bug Report: No bug has been found in my database testing.



Bug Reporting

# SL 05

Issue:

Reproducing Steps:

Environment

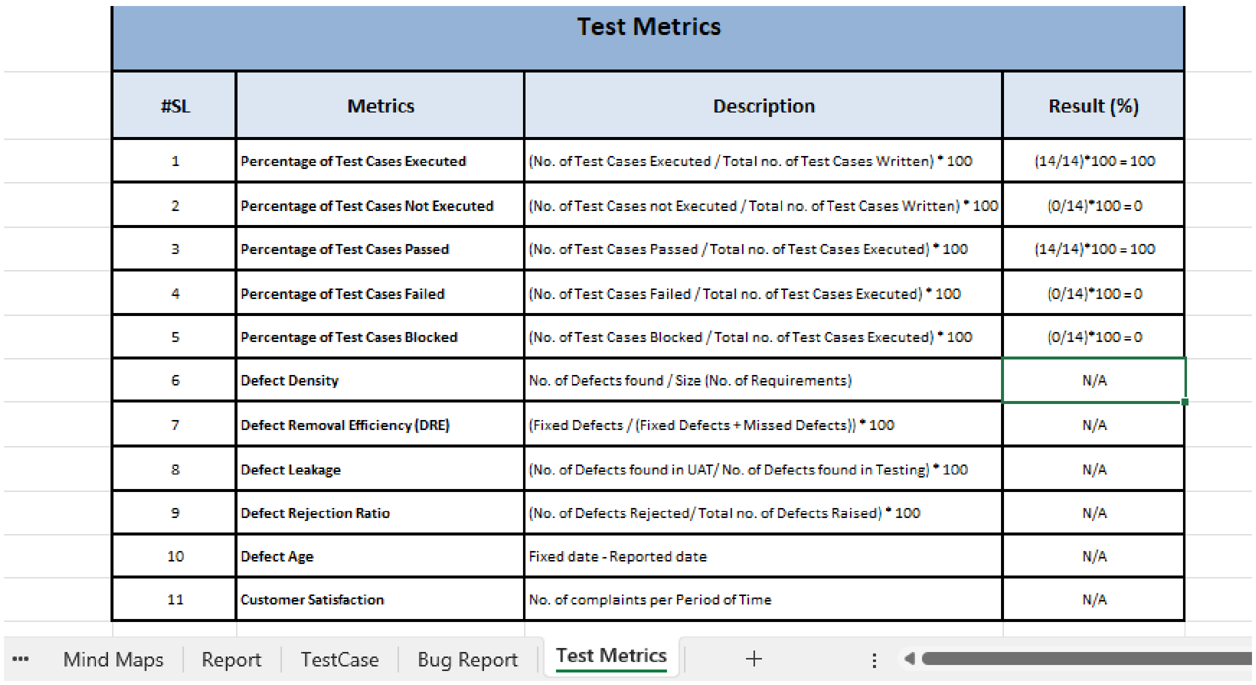
Module: Database Testing

Severity: P1

Screenshot:

Responsible QA:Mir Mahadi Hossain

Test Metrics:


Mind Maps
Report
TestCase
Bug Report
Test Metrics
+

Test Metrics

#SL	Metrics	Description	Result (%)
1	Percentage of Test Cases Executed	(No. of Test Cases Executed / Total no. of Test Cases Written) * 100	(14/14)*100 = 100
2	Percentage of Test Cases Not Executed	(No. of Test Cases not Executed / Total no. of Test Cases Written) * 100	(0/14)*100 = 0
3	Percentage of Test Cases Passed	(No. of Test Cases Passed / Total no. of Test Cases Executed) * 100	(14/14)*100 = 100
4	Percentage of Test Cases Failed	(No. of Test Cases Failed / Total no. of Test Cases Executed) * 100	(0/14)*100 = 0
5	Percentage of Test Cases Blocked	(No. of Test Cases Blocked / Total no. of Test Cases Executed) * 100	(0/14)*100 = 0
6	Defect Density	No. of Defects found / Size (No. of Requirements)	N/A
7	Defect Removal Efficiency (DRE)	(Fixed Defects / (Fixed Defects + Missed Defects)) * 100	N/A
8	Defect Leakage	(No. of Defects found in UAT/ No. of Defects found in Testing) * 100	N/A
9	Defect Rejection Ratio	(No. of Defects Rejected/ Total no. of Defects Raised) * 100	N/A
10	Defect Age	Fixed date - Reported date	N/A
11	Customer Satisfaction	No. of complaints per Period of Time	N/A

Conclusion Note:

- The College Library Management System is a comprehensive database project designed to enhance the efficiency and functionality of a college library. With its well-defined objectives, key features, and numerous benefits, the system aims to streamline the library processes, making it easier for librarians to manage the library's resources and for students to access and borrow books.
- By utilizing an E-R diagram, the system's database structure is intelligently organized, ensuring effective data management and retrieval. The creation and utilization of tables provide a structured approach to store and maintain essential information about books, students, borrowing history, fines, and more.
- The implementation of the system facilitates easy insertion of data, allowing librarians to update book information, add new books to the collection, and maintain accurate records of students and their borrowing activities.
- The ability to create and execute queries empowers the librarians to generate valuable insights into the library's functioning. These queries help track book availability, identify popular titles, manage fines, and analyze user behavior, enabling data-driven decision-making and resource optimization.
- In conclusion, the College Library Management System serves as a valuable tool in modernizing and optimizing library operations. By automating various tasks and centralizing information, the system not only saves time and resources but also enhances the overall user experience for both librarians and students. It promotes efficient resource management, timely notifications, and data-driven decision-making, contributing to a more effective and well-managed college library.