

**IN
PARTNERSHIP
WITH
PLYMOUTH
UNIVERSITY**

Name: GPDCM Jayasekara

Student Reference Number: 10707085

Module Code: ISAD253SL

Module Name: Databases

Coursework Title: Group Assignment

Deadline Date: Monday, 4 January 2021

Member of staff responsible for coursework:
Mr. Naji Saravanabavan

Programme: BSc (Honours) Software Engineering / BSc (Honours) Computer Networks /
BSc (Honours) Computer Security

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

Member Activity	Member 01 ID: 10707085	Member 02 ID: 10707385	Member 03 ID: 10707372	Member 04 ID: 10707241	Member 05 ID: 10707052	Member 06 ID: 10707053
Book Table						
Borrower Table						

Copy Table						
Issue-Info Table						
Publisher Table						
Library Table						
Staff & Authentication System						

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: **I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.**

Signed:

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software.....

Overall mark _____ % Assessors Initials _____ Date _____

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

LIBRARY MANAGEMENT SYSTEM

Table of Contents

1.	SECTION 01	6
❖	INTRODUCTION TO THE SCENARIO LIBRARY MANAGEMENT SYSTEM	6
❖	EXTENDED ENTITY RELATIONSHIP DIAGRAM	7
❖	ASSUMPTIONS MADE REGARDING THE EERD	8
❖	RELATIONAL MAPPING.....	10
❖	NORMALIZATION	11
1)	BOOK TABLE.....	11
2)	BORROWER TABLE.....	12
3)	COPY TABLE	12
4)	ISSUEINFO TABLE.....	13
5)	STAFF TABLE	14
6)	LIBRARY TABLE	15
7)	PUBLISHER TABLE	16
❖	DATA DICTIONARY	18
2.	SECTION 02.....	21
❖	CREATE TABLES AND CONSTRAINTS	21
1)	BOOK TABLE.....	21
2)	BORROWER TABLE.....	22
3)	COPY TABLE	24
1.	Create table and use constraints.....	24
4)	ISSUE-INFO TABLE	25
1.	Create table and use constraints.....	25
5)	STAFF TABLE	26
6)	PUBLISHER TABLE	29
7)	LIBRARY TABLE	30
❖	DB DIAGRAMS	31
❖	SAMPLE RECORDS	32
1)	BOOK TABLE.....	32
3)	COPY TABLE	35
1.	Insert sample data	35
4)	ISSUE-INFO TABLE	36

1.	Insert sample data	36
5)	STAFF TABLE	37
6)	PUBLISHER TABLE	38
7)	LIBRARY TABLE	39
3.	SECTION 03	40
❖	TRIGGERS	40
1)	BOOK TABLE.....	40
2)	BORROWER TABLE.....	42
3)	COPY TABLE	45
1.	After Update Trigger.....	45
1.	After Delete Trigger.....	46
4)	ISSUE-INFO TABLE	47
1.	After Update Trigger.....	47
2.	After Delete Trigger.....	48
5)	STAFF TABLE	49
6)	PUBLISHER TABLE	52
7)	LIBRARY TABLE	55
❖	DATABASE VIEWS	57
1)	BOOK TABLE.....	57
2)	BORROWER TABLE.....	58
3)	COPY TABLE	60
1.	Create View	60
2.	Create View	60
4)	ISSUE-INFO TABLE	61
1.	Create View	61
2.	Create View	61
5)	STAFF TABLE	62
6)	PUBLISHER TABLE	64
7)	LIBRARY TABLE	66
❖	USER DEFINED FUNCTIONS	67
1)	BOOK TABLE.....	67
2)	BORROWER TABLE.....	68
3)	COPY TABLE	70
1.	Create SQL Scalar Functions.	70

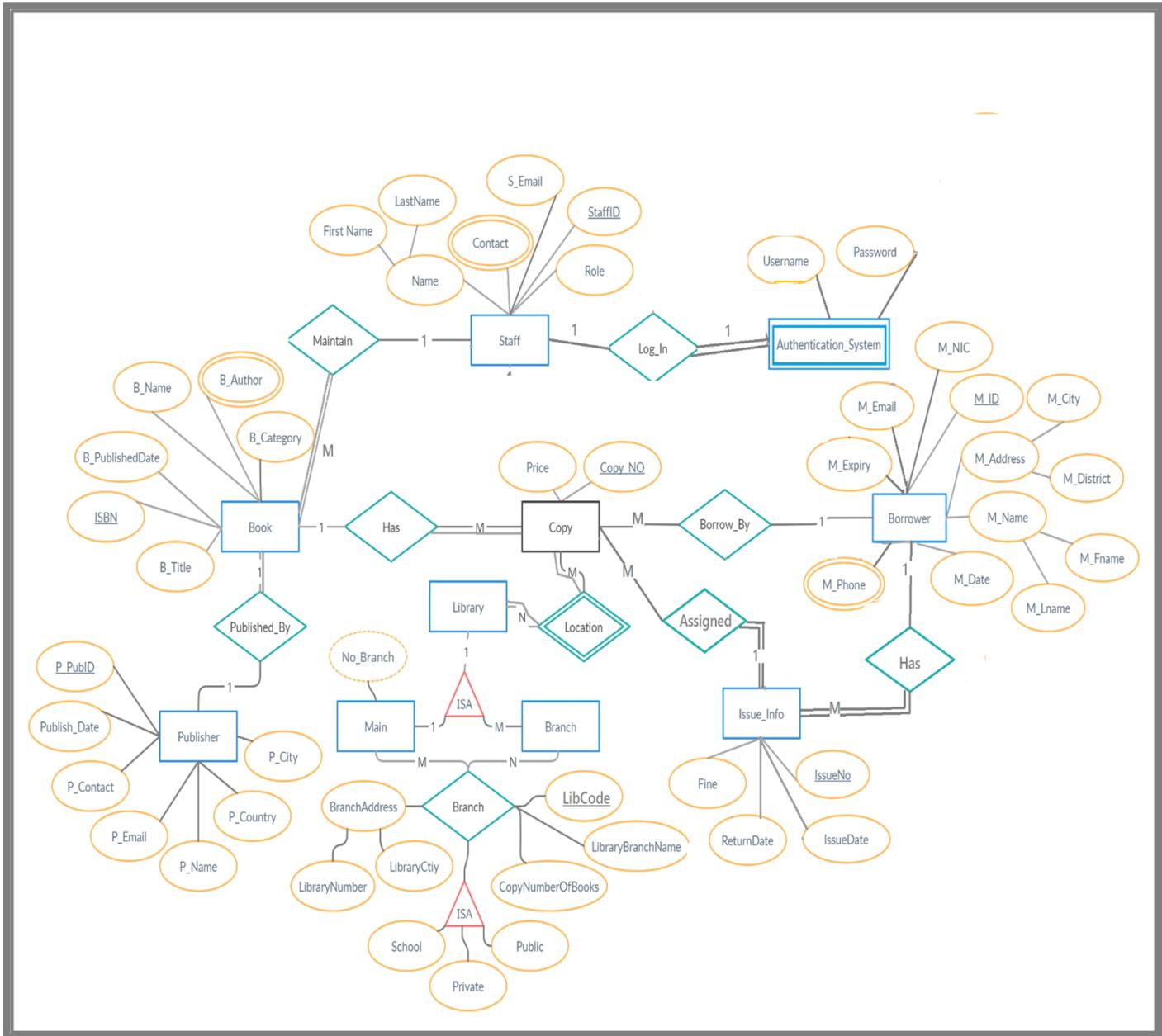
2.	Create SQL Scalar Functions.....	70
4)	ISSUE-INFO TABLE	71
1.	Create SQL Scalar Functions.....	71
2.	Create SQL Scalar Functions.....	71
5)	STAFF TABLE	73
6)	PUBLISHER TABLE	75
❖	STORED PROCEDURES	76
1)	BOOK TABLE.....	76
2)	BORROWER TABLE.....	77
3)	COPY TABLE	79
1.	Select Stored Procedure.....	79
2.	Update Stored Procedure.....	82
4)	ISSUE-INFO TABLE	83
1.	Select Stored Procedure.....	83
2.	Update Stored Procedure.....	85
5)	STAFF TABLE	86
6)	PUBLISHER TABLE	88
4.	SECTION 04	89
❖	CRITICAL EVALUATION	89
❖	FUTURE IMPLEMENTATION.....	90

1. SECTION 01

❖ INTRODUCTION TO THE SCENARIO LIBRARY MANAGEMENT SYSTEM

A library service wants to create a database to store details of its libraries, books, borrowers, publishers, and staff. Details include the following: A book has a unique ISBN number, a title, published date, book name, category, and authors. The library service may own several copies of a given book, each of which is located in one of the service's libraries. A given library contains many books, and in order to distinguish different copies of the same book a library assigns a different copy-number to each of its copies of a given book; the price that was paid for each copy is also recorded. For every book which has been issued with a unique issue number along with a fine, issue date and return date. If a borrower does not return the book in a duration of 14 days then, a fine is calculated based on the extra days. A borrower has a name, a unique ID code along with a unique NIC, email, phone, district, city, membership date, membership expire date. A borrower can have many books on loan, but each copy of a book can only be on loan to one borrower. A borrower could borrow the same book on several occasions, but it is assumed that each such loan will take place on a different date. Each publisher has a unique publisher ID, publisher name, published date and a published country. Each publisher has a unique contact number, and each publisher publishes only one book. Every library has a unique library code along with a library branch name, branch address and library types, the number of copies issued to a library is also recorded and is either a main library or a branch library. A main library may have zero or more branch libraries and every branch library is a branch of exactly one main library. Every Staff member has a staff id. Every staff member are not administrators. Admin can view all the information about staff members. Only admin can enter the details of staff members into database, but staff can't do their own. An admin can add new staff members, delete, grant/revoke access to database, track borrowers' details and maintain details of resources available in premises. Only admins can login into the systems with their own passwords and username.

❖ EXTENDED ENTITY RELATIONSHIP DIAGRAM



❖ **ASSUMPTIONS MADE REGARDING THE EERD**

- A book can have one or more authors (B_Author)
- The name published date and category of the book is also additionally recorded
- A borrower can have one or more phone numbers (M_Phone)
- A borrower's address (M_Address) consist of 2 parts city (M_City) & district (M_District), his\hers address can be null.
- A borrower's name (M_Name) consist of 2 parts first name (M_Fname) & second name (M_Lname).
- A borrower's unique NIC (M_NIC), unique email (M_Email), start of membership date (M_Date) and membership expiry date (M_Expiry) is also recorded as it provides clear information and to properly identify the borrower.
- Added IssueInfo table to save issued books data.
- An IssueInfo table attributes are Issue number (IssueNo- which is increment automatically), Fine, Issue date (IssueDate), Return date (ReturnDate).
- A Library can have a main with many branches or no branches however a branch cannot be created without a main (Library) (main) (branch)
- Each library should have a name which is not null (BranchName)
- Further the number of copies received to each branch is recorded (CopyNumber)
- Assuming that there are many types of libraries the following attributes are additionally created (public)(private)(school) are broken down by the relationship from the (branch)
- (BranchAddress) is created assuming that each library has an address which is not null since it is compulsory.
- After considering the above scenario we have added two additional attributes of published city (Pub_City) and publisher E-mail (Pub_Email) inorder to provide more information about the publisher.
- Publisher ID (Pub_ID) is the primary key.
- We assume that publisher contact (Pub_Contact) and publisher e-mail (Pub_Email) addresses are not multi-valued attributes and can record only one value for each of the publisher data input
- Staff entity has attributes named, Name, Contact, Email, StaffID (primary key which is set to incremented) and Staff Role
- Staff members are required for library to check status of resources.
- StaffID is the primary key for staff members as it helps to uniquely identify each single member.

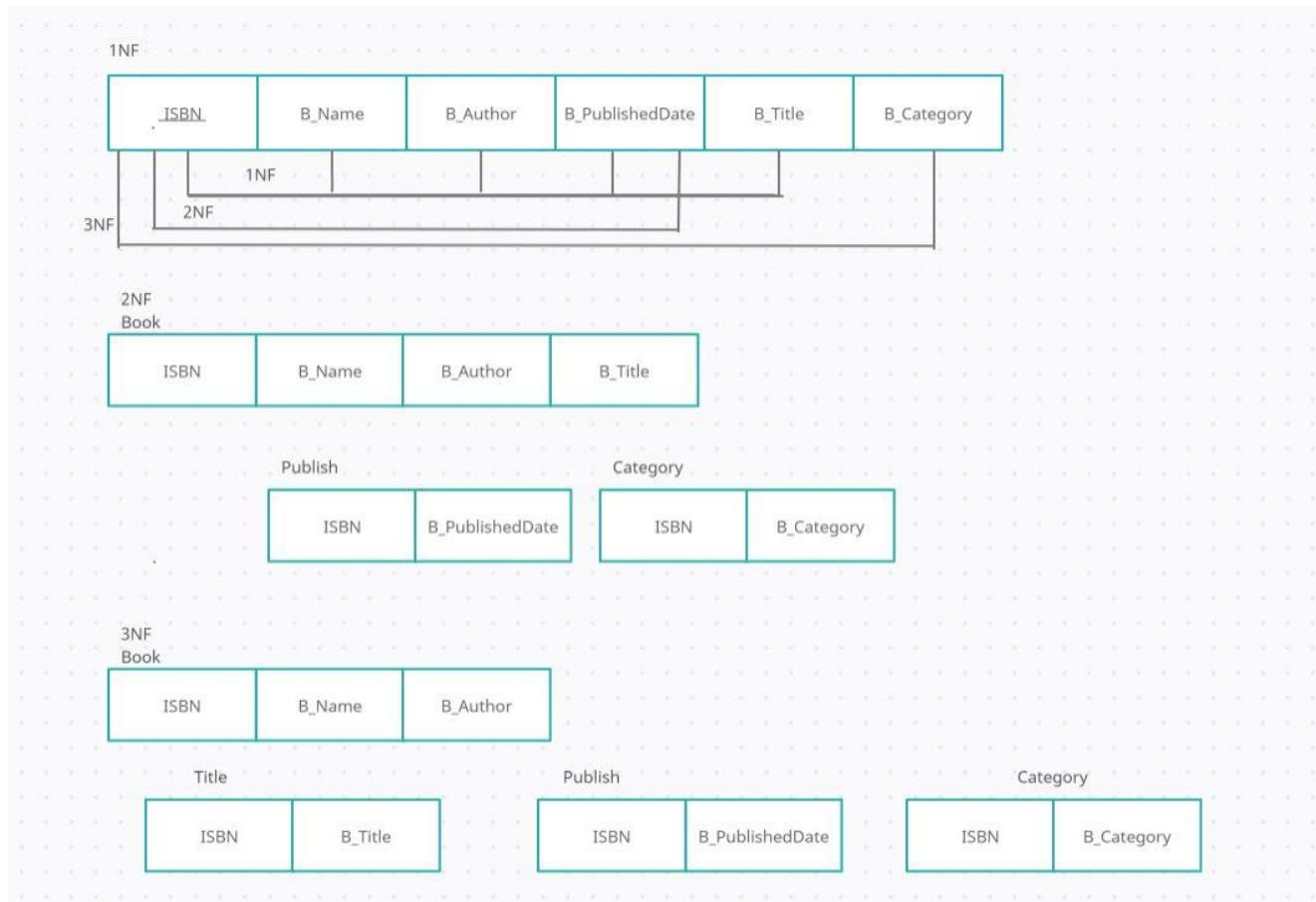
- Staff members who want log into the system are required to provide the valid username and password. Using username and password can prevent unauthorized parties to login the system.
- Staff members who have thier role set to admin are considered as administrators and only them can login the system.
- Name of the staff members are breaked into first and last name in separate fields to identify easily.
- Contact of members are added within a multivalued attribute because a member can have one or more contact numbers.
- Through successful authentication system, anyone can see the details of resources as well as manage them too.
- By checking the details of resources, library can make their own analysis as it will help them make decisions as well.

❖ RELATIONAL MAPPING



❖ NORMALIZATION

1) BOOK TABLE



2) BORROWER TABLE

1NF

Borrower

M_ID	M_fname	M_lname	M_NIC	M_Email	M_Phone	M_District	M_City	M_Date	M_Expiry
------	---------	---------	-------	---------	---------	------------	--------	--------	----------

2NF

Borrower

M_ID	M_NIC	M_Email	M_Phone	M_Date	M_Expiry

3NF

Borrower Personal

M_ID	M_NIC

Borrower Contact

M_ID	M_Email	M_Phone

Borrower Membership

M_ID	M_Date	M_Expiry

3) COPY TABLE

1NF

Copy

CopyNo	Price	Copy_ISBN

3NF

Copy

CopyNo	Copy_ISBN

CopyPrice

CopyNo	Price

4) ISSUEINFO TABLE

1NF

IssueInfo

<u>IssueNo</u>	Fine	IssueDate	ReturnDate	Member_ID	B_ISBN	CopyNo
----------------	------	-----------	------------	-----------	--------	--------

2NF

IssueInfo

<u>IssueNo</u>	Fine	IssueDate	ReturnDate
----------------	------	-----------	------------



```
graph TD; A[IssueNo] --> B[Fine]; A --> C[IssueDate]; A --> D[ReturnDate]
```

3NF

IssueFine

<u>IssueNo</u>	Fine
----------------	------



```
graph TD; A[IssueNo] --> B[Fine]
```

IssueInfo

<u>IssueNo</u>	IssueDate	ReturnDate
----------------	-----------	------------



```
graph TD; A[IssueNo] --> B[IssueDate]; A --> C[ReturnDate]
```

5) STAFF TABLE

It's required to present the normalization up to third normal form 3NF. To proceed we need to satisfy first normal form 1NF, second normal form 2NF and then 3NF. Tables in database are believed to be in 3NF form. When normalize the contents that able to keep the design optimized.

Staff Table before normalization

StaffID	Name	Contact	Email	Joined Date	Staff Role	Credentials
---------	------	---------	-------	-------------	------------	-------------

First normal form (1NF)

<u>StaffID</u>	FirstName	LastName	Contact	Email	Joined Date	Staff Role	Username	Password
----------------	-----------	----------	---------	-------	-------------	------------	----------	----------

For first normal formal, changes that made are, setting Staff ID as a primary key and for the name, separating Name field into two fields as, First Name and Last Name.

And break credentials into two parts called Username and password.

Second Normal Form (2NF)

In 2nd Normal form, the attributes without keys (non key attributes) should be completely dependent on primary key.

Initially, I created another two tables for staff members and a sperate table for insert credentials for authorized staff. With this, it will help to associate and relate data within the foreign key as well.

Staff Table

<u>StaffID</u>	FirstName	LastName	Contact	Email	Joined Date	Staff Role
----------------	-----------	----------	---------	-------	-------------	------------

Authentication System Table

Staff ID - FK	User Name	Password
---------------	-----------	----------

Now, with using 2NF there cannot be repeating fields and no partial dependency on primary key as well.

Third Normal Form (3NF)

In third normal form, we need to discard the non key attributes that do not maintain complete dependency on primary key. But by observing above table there cannot be found those kind of attributes, therefore, final 3NF result will be, same as 2NF form,

Staff Table

<u>StaffID</u>	FirstName	LastName	Contact	Email	Joined Date	Staff Role

Authentication System Table

Staff ID - FK	User Name	Password
---------------	-----------	----------

6) LIBRARY TABLE

Library

1nf

<u>libcode</u>	LibraryType	BranchName	BranchType	city	LibNumber	copynumber
----------------	-------------	------------	------------	------	-----------	------------

2nf

<u>libcode</u>	BranchName	city	LibNumber

3nf

Name details

<u>libcode</u>	BranchName

address

<u>libcode</u>	City	libNumber

7) PUBLISHER TABLE

Currently the table is in 0NF. So, it has to be converted to 1NF.

Pub_ID	Pub_Name	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email

When converting to 1NF all the multivalued and composite attributes should be removed. For the publisher table to be in 1NF it should fulfill the following requirements.

- It should only have single attributes.
- All the columns in the name should have unique names.
- Values stored in the column should be of the same domain.
- The order of data does not matter

So the above table fulfills all the requirements which means that the publisher table is already in 1NF.

Pub_ID	Pub_Name	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email

Next to convert it into 2NF publisher table should fulfill the below two requirements of,

- It should be in 1NF
- And should not have any partial dependencies.

Since the publisher satisfies the above two requirements it is in 2NF which is shown below.

Pub_ID	Pub_Name	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email

After converting the publisher table to 2NF we must further convert it to 3NF. To do so our table should be comprised of,

- Table should be in 2NF
- It does not have transitive dependencies.

So by considering the above facts it shows that the publisher table had already been normalized. So the final publisher table would be as follows.

Pub_ID	Pub_Name	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email

❖ DATA DICTIONARY

Book

Attribute Name	Data Type	Required	Field Size
ISBN	varchar		5
B_Name	varchar	NOT NULL	50
B_Author	varchar	NOT NULL	50
B_PublishedDate	date		
B_Title	varchar		50
B_Category	varchar	NOT NULL	50

Borrower

Attribute Name	Data Type	Required	Field Size
M_ID	varchar	NOT NULL	4
M_NIC	varchar	NOT NULL	12
M_Fname	varchar	NOT NULL	15
M_Lname	varchar	NOT NULL	15
M_Email	varchar	NOT NULL	50
M_Phone	int	NOT NULL	
M_District	varchar	NULL	20
M_City	varchar	NULL	20
M_Date	date	NOT NULL	
M_Expiry	date	NOT NULL	

Copy

PK / FK	Attribute Name	Data Type	Required	Field Size
PK	CopyNo	varchar	NOT NULL	10
FK	Copy_ISBN	varchar	NOT NULL	5
	Price	float	NOT NULL	

IssueInfo

PK / FK	Attribute Name	Data Type	Required	Field Size
PK	IssueNo	int	NOT NULL	
FK	B_ISBN	varchar	NOT NULL	5
FK	CopyNo	varchar	NOT NULL	10
FK	Member_ID	varchar	NOT NULL	4
	Fine	float	NULL	
	IssueDate	date	NOT NULL	
	ReturnDate	date	NOT NULL	

Publisher table

Attribute Name	Data Type	Required	Field Size
Pub_ID	Varchar	NOT NULL	5
Pub_Name	Varchar	NULL	50
Published_BookID	Varchar	NOT NULL	5
Pub_Country	Varchar	NULL	50
Pub_City	Varchar	NULL	50
Pub_Date	Date	NULL	
Pub_Contact	Varchar	UNIQUE	50
Pub_Email	Varchar	NULL	50

LIBRARY TABLE

PK/UK	Field name	Data type	Field length	caption	Description
Primary key	libcode	char	20		Libcode auto generated one after the other
	LibraryType	varchar	50	main/branch	The users should enter only these options
	BranchName	varchar	50		Name of the library
	BranchType	varchar	50	Private/public/school	The user should only enter these options

	city	varchar	20		The city of the library should be mentioned
Unique key	LibNumber	Varchar	20		The number of the building
	copynumber	Varchar	20		The number of copies distributed for each library is recorded

STAFF TABLE

Attribute	Data Type	Required	Field Size
FirstName	VARCHAR	NOT NULL	50
LastName	VARCHAR	NOT NULL	50
Contact	INT	NOT NULL	
Email	VARCHAR		30
StaffID	INT	NOT NULL	
StaffRole	VARCHAR	NOT NULL	20

2. SECTION 02

❖ CREATE TABLES AND CONSTRAINTS

1) BOOK TABLE

➤ TABLE CREATION AND CONSTRAINT

The number of attributes that I have added are five attributes where the ISBN is the Primary Key and I have used only a Constraint for the Primary Key.

```
*****TABLE CREATION*****  
  
CREATE TABLE Book(  
    ISBN varchar(5),  
    B_Name varchar (50) NOT NULL,  
    B_Author varchar (50) NOT NULL,  
    B_PublishedDate DATE,  
    B_Title varchar (50),  
    B_Category varchar (50) NOT NULL,  
    CONSTRAINT Pk_isbn PRIMARY KEY (ISBN),  
);  
  
SELECT * FROM Book;
```

2) BORROWER TABLE

DATABASE CREATION FOR BORROWER

```
CHAMOTH.sql - (Lo...th Madushan (52))  ✎ ×
/******DATABASE CREATION*****/
/* DATABASE CREATION*/
CREATE DATABASE BORROWER;
USE BORROWER;
```

- . In this a database named Borrower is created and used to maintain library member information.

TABLE CREATION FOR BORROWER

```
/******TABLE CREATION*****
/*TABLE CREATION BORROWER*/
CREATE TABLE Borrower (
M_ID varchar (4),
M_Fname varchar (15) NOT NULL,
M_Lname varchar (15) NOT NULL,
M_Email varchar (50) NOT NULL,
M_Phone int NOT NULL,
M_NIC varchar (12) NOT NULL,
M_District varchar (20),
M_City varchar (20),
M_Date date NOT NULL,
M_Expiry date NOT NULL,
CONSTRAINT pk_mid PRIMARY KEY (M_ID),
CONSTRAINT uq_mnic UNIQUE (M_NIC),
CONSTRAINT uq_memail UNIQUE (M_Email),
CONSTRAINT uq_mphone UNIQUE (M_Phone),
);
/*TABLE CREATION NEW MEMBER*/
CREATE TABLE new_member (
m_id varchar (4),
m_data varchar (100),
);
```

- Two tables have been created one is the Borrower where the borrowers details are stored & the new_member where all the activities of the information is stored.

Borrower Table Execution

M_ID	M_Fname	M_Lname	M_Email	M_Phone	M_NIC	M_District	M_City	M_Date	M_Expiry
------	---------	---------	---------	---------	-------	------------	--------	--------	----------

New_member Table Execution

m_id	m_data
------	--------

CONSTRAINTS FOR BORROWER TABLE

```
CONSTRAINT pk_mid PRIMARY KEY (M_ID),  
CONSTRAINT uq_mnic UNIQUE (M_NIC),  
CONSTRAINT uq_email UNIQUE (M_Email),  
CONSTRAINT uq_phone UNIQUE (M_Phone),
```

- The Primary Key is the M_ID as it can be used to identify the borrower uniquely and shall be used as the primary key in the table.
- NIC is considered as a unique and identification of the borrower.
- The email/phone address is also considered as unique as a borrower's email/phone address cannot be the same of another borrower's email/phone address.

3) COPY TABLE

1. Create table and use constraints.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'LibraryManagementSystem' database and its tables like 'Book', 'Borrower', and 'Copy'. The central pane displays a SQL script for creating the 'Copy' table:

```
CREATE TABLE Copy
(
    Copy_ISBN varchar(5) NOT NULL,
    CopyNo varchar(10),
    Price float NOT NULL,
    CONSTRAINT pk_CopyNo PRIMARY KEY (CopyNo), /*PRIMARY KEY*/
    CONSTRAINT fk_ISBN FOREIGN KEY
        (Copy_ISBN) REFERENCES Book(ISBN) /*FOREIGN KEY*/
)
SELECT * FROM Copy;
```

The 'Messages' tab shows the command completed successfully with a completion time of 2021-01-01T16:22:08.9199577+05:30. The bottom status bar indicates 'Query executed successfully.' and '0 rows'.

In the second pane, a 'SELECT * FROM Copy;' query is run, resulting in 0 rows. The results grid shows columns: Copy_ISBN, CopyNo, and Price.

4) ISSUE-INFO TABLE

1. Create table and use constraints.

- This ‘IssueInfo’ table is created to store issued book details.
- This table have a Primary Key and three Foreign Keys.
- There are seven columns in this IssueInfo table, such as IssueNo, B_ISBN, CopyNo, Member_ID, Fine, IssueDate &ReturnDate.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'Tables' node which contains 'IssueInfo'. The central pane displays the T-SQL code for creating the 'IssueInfo' table:

```
/*
*****CONSTRAINTS*****
*/
CREATE TABLE IssueInfo
(
    IssueNo int IDENTITY(1,1),          /*AUTO INCREMENT*/
    B_ISBN varchar(5) NOT NULL FOREIGN KEY (B_ISBN) REFERENCES Book(ISBN), /*FOREIGN KEY*/
    CopyNo varchar(10) NOT NULL,
    Member_ID varchar(4) NOT NULL FOREIGN KEY (Member_ID) REFERENCES Borrower(M_ID), /*FOREIGN KEY*/
    Fine float,
    IssueDate DATE NOT NULL,
    ReturnDate DATE NOT NULL,
    CONSTRAINT pk_IssueNo PRIMARY KEY (IssueNo), /*PRIMARY KEY*/
    CONSTRAINT fk_CopyNo FOREIGN KEY
        (CopyNo) REFERENCES Copy(CopyNo), /*FOREIGN KEY*/
);
SELECT * FROM IssueInfo;
```

The 'Results' tab at the bottom shows the output of the 'SELECT *' query, which is empty (0 rows). The status bar at the bottom indicates 'Query executed successfully.'

5) STAFF TABLE

Create table Staff

Creating a database named Staff to start off creating tables and inserting data,

SQLQuery1.sql - LA...1SS03\Dasitha (/4))'

```
CREATE DATABASE Staff;
```

After creating database Staff and USE statement, created a table called StaffTable to insert details of all staff members considering StaffID as a primary key.

```
USE Staff;

CREATE TABLE StaffTable(
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    Contact int NOT NULL,
    Email varchar(30),
    StaffID int IDENTITY(1,1) NOT NULL,
    StaffRole varchar(20) NOT NULL,
    CONSTRAINT pk_staffid PRIMARY KEY (StaffID),
    CONSTRAINT u_email UNIQUE (Email),
    CONSTRAINT u_fullname UNIQUE (FirstName,LastName)
);
```

Only email is set to null field, as some if the staff members are not posses an email address. StaffID is set to be increment by 1 as a effective way to insert the data easily and avoid the integrity.

Email and both First Name and Last Name are considered as unique, in case to identify easily with. Also JoinedDate also added to show date that joined.

```

CREATE TABLE StaffTable(
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    Contact int NOT NULL,
    Email varchar(30),
    StaffID int IDENTITY(1,1) NOT NULL,
    StaffRole varchar(20) NOT NULL,
    Joined_Date date NOT NULL,
    CONSTRAINT pk_staffid PRIMARY KEY (StaffID),
    CONSTRAINT u_email UNIQUE (Email),
    CONSTRAINT u_fullname UNIQUE (FirstName,LastName)
);

SELECT * FROM StaffTable

```

.00 % ▶

Results	Messages					
FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date

After creating table StaffTable, another table called AuthSysTable is also created within Staffid, Username and Passwords as an attributes.

```

CREATE TABLE AuthSysTable(
    StaffID INT NOT NULL,
    Username varchar(20) UNIQUE NOT NULL,
    Password varchar(20) UNIQUE NOT NULL,
);

```

Only staff members who are admin, their credentials are inserted into this table as well with their own staff id. Added a foreign key constraint to Staff ID

```

ALTER TABLE AuthSysTable
ADD CONSTRAINT fk_staffid FOREIGN KEY (StaffID) REFERENCES StaffTable (StaffID)

```

Create Table New Staff Members

After inserting data, I created another table for new members to identify the members who joined quickly by using the statements,

```
CREATE TABLE newmembers(
    Staff_Id int NOT NULL,
    joineddate date NOT NULL);
```

6) PUBLISHER TABLE

In order to create the publisher table, first of all the database must be created. So, the database would be Library

1. Creation of database

```
CREATE DATABASE Library;

USE Library;
```

100 % ▶

Messages

Commands completed successfully.

Completion time: 2021-01-03T18:30:41.7549910+05:30

2. Creation of the table with constraints

```
CREATE TABLE Publisher (
    Pub_ID varchar (5),
    Pub_Name varchar (50),
    Published_BookID varchar (5) FOREIGN KEY (Published_BookID) REFERENCES dbo.Book (ISBN), /* ISBN USED BY BOOK TABLE TO CREATE A FOREIGN KEY */
    Pub_Country varchar (50),
    Pub_City varchar (50) DEFAULT 'Colombo', /* DEFAULT CONSTRAINT USED TO FILL THE ROW AUTOMATICALLY */
    Pub_Date date,
    Pub_Contact char (10),
    Pub_Email varchar (50),
    CONSTRAINT pk_id PRIMARY KEY (Pub_ID), /* PRIMARY KEY OF THE TABLE */
    CONSTRAINT uk_cntNo UNIQUE (Pub_Contact), /* UNIQUE CONSTRAINT IS USED AS CONTACT NUMBERS DIFFER */
);
```

100 % ▶

Messages

Commands completed successfully.

Completion time: 2021-01-03T16:11:19.2289094+05:30

After creating and executing the publisher table the following result of table occurs.

```
SELECT * FROM Publisher; /* DISPLAYS THE PUBLISHER TABLE */
```

100 % ▶

Results □ Messages

Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email

7) LIBRARY TABLE

```
CREATE TABLE library (libcode char (20) not null,
LibraryType varchar (50),
BranchName varchar (50) not null,
BranchType varchar(50) not null,
City varchar (20) not null,
LibNumber varchar (20) not null,
CopyNumber varchar (20) not null,
CONSTRAINT pk_id PRIMARY KEY (libcode),
CONSTRAINT uk_address UNIQUE (Libnumber),);
```

50 %

	libcode	LibraryType	BranchName	BranchType	City	LibNumber	CopyNumber
1	001	main	nex	public	colombo	12/45	63
2	002	Branch	IBT	school	colombo	11/45	05
3	003	Branch	NSBM	school	homagama	31/45	03
4	004	branch	nex	public	kandy	15/45	33
5	005	main	NSBM	school	colombo	18/45	43
6	006	branch	norton	private	colombo	10/45	72
7	007	branch	leeds	school	Galle	60/45	92
8	008	main	weed	private	colombo	05/45	70
9	009	branch	team it	public	Thalawathugoda	54/45	32
10	010	branch	oxford	public	Thalawathugoda	04/45	27

The above are the sql statements that were used to create this table which includes a primary key (libcode) and a unique key (LibNumber). Further the libcode is the unique ID given to each library, the LibNumber is a unique key since there cannot be two or more libraries at the same place. Other columns like LibraryType is to know if the library is a main or branch, BranchName stores the name of the library. Where as city stores the city of the library and libNumber stores the building number of the library, these two could be known as the BranchAddress however a column like that isn't created since we cannot put two values in one column. Apart from that the last column (copynumber) stores the number of copies received for each library.

❖ DB DIAGRAMS



❖ SAMPLE RECORDS

1) BOOK TABLE

DATA INSERTION TO THE TABLE

Data insertion are shown below with the results.

```
INSERT INTO Book
Values('B001', 'Harry Potter ', 'J. K. Rowling', '1997', 'and the Philosophers Stone', 'Novel');

INSERT INTO Book
Values('B002', 'Harry Potter', 'J. K. Rowling', '2007', 'and the Deathly Hallows', 'Novel');

INSERT INTO Book
Values('B003', 'Harry Potter', 'J. K. Rowling', '2016', 'and the Cursed Child', 'Novel');

INSERT INTO Book
Values('B004', 'Harry Potter', 'J. K. Rowling', '2000', 'and the Goblet of Fire ', 'Novel');

INSERT INTO Book
Values('B005', 'Harry Potter', 'J. K. Rowling', '2003', 'and the Order of the Phoenix ', 'Novel');

INSERT INTO Book
Values('B006', 'Harry Potter', 'J. K. Rowling', '2003', 'and the Chamber of Secrets ', 'Novel');

INSERT INTO Book
Values('B007', 'Harry Potter', 'J. K. Rowling', '1999', 'and the Prisoner of Azkaban ', 'Novel');

INSERT INTO Book
Values('B008', 'Cinderella', 'Daisy Fisher', '1981', '', 'Fairy tale');

INSERT INTO Book
Values('B009', 'Dracula', 'Bram Stoker', '1897', '', 'Novel');

INSERT INTO Book
Values('B010', 'House of leaves', 'Mark Z. Danielewski', '2000', '', 'Novel');

INSERT INTO Book
Values('B011', 'Wonder Woman:', 'Leigh Bardugo', '2017', 'Warbringer', 'Fiction');
```

```

    □ INSERT INTO Book
    | Values('B012', 'The Chronicles', 'C.S.Lewis', '1950', 'Narnia', 'Novel');

    □ INSERT INTO Book
    | Values('B013', 'Twilight', 'Stephenie Meyer', '2006', 'New moon', 'Novel');

    □ INSERT INTO Book
    | Values('B014', 'A Promised Land', 'Barack Obama', '2020', '', 'Autobiography');

    □ INSERT INTO Book
    | Values('B015', 'Goosebumps', 'R. L. Stine', '1993', 'The Haunted Mask', 'Horror fiction');

    □ INSERT INTO Book
    | Values('B016', 'Goosebumps', 'R. L. Stine', '1992', 'Welcome to Dead House', 'Horror fiction');

    □ INSERT INTO Book
    | Values('B017', 'Goosebumps', 'R. L. Stine', '1993', 'Night of the Living Dummy', 'Horror fiction');

    □ INSERT INTO Book
    | Values('B018', 'White House:', 'Amy Bloom', '2018', 'A Novel', 'Historical fiction');

    □ INSERT INTO Book
    | Values('B019', 'Twilight', 'Stephenie Meyer', '2007', 'Eclipse', 'Novel');

    □ INSERT INTO Book
    | Values('B020', 'Twilight', 'Stephenie Meyer', '2008', 'Breaking Dawn', 'Novel');

    SELECT * FROM Book;

```

TABLE RESULT

The results are shown as below after inserting to the table.

100 %

	ISBN	B_Name	B_Author	B_PublishedDate	B_Title	B_Category
1	B001	Harry Potter	J. K. Rowling	2021-01-03	and the Philosophers Stone	Novel
2	B002	Harry Potter	J. K. Rowling	2021-01-03	and the Deathly Hallows	Novel
3	B003	Harry Potter	J. K. Rowling	2021-01-03	and the Cursed Child	Novel
4	B004	Harry Potter	J. K. Rowling	2021-01-03	and the Goblet of Fire	Novel
5	B005	Harry Potter	J. K. Rowling	2021-01-03	and the Order of the Phoenix	Novel
6	B006	Harry Potter	J. K. Rowling	2021-01-03	and the Chamber of Secrets	Novel
7	B007	Harry Potter	J. K. Rowling	2021-01-03	and the Prisoner of Azkaban	Novel
8	B008	Cinderella	Daisy Fisher	2021-01-03		Fairy tale
9	B009	Dracula	Bram Stoker	2021-01-03		Novel
10	B010	House of l...	Mark Z. Da...	2021-01-03		Novel
11	B011	Wonder ...	Leigh Bard...	2021-01-03	Warbringer	Fiction
12	B012	The Chro...	C.S.Lewis	2021-01-03	Narnia	Novel
13	B013	Twilight	Stephenie ...	2021-01-03	New moon	Novel
14	B014	A Promise...	Barack Ob...	2021-01-03		Autobiogr...
15	B015	Approved	R. L. Stine	2021-01-03	The Haunted Mask	Horror fict...
16	B016	Goosebu...	R. L. Stine	2021-01-03	Welcome to Dead House	Horror fict...
17	B017	Goosebu...	R. L. Stine	2021-01-03	Night of the Living Dummy	Horror fict...
18	B018	White Ho...	Amy Bloom	2021-01-03	A Novel	Historical ...
19	B019	Twilight	Stephenie ...	2021-01-03	Eclipse	Novel
20	B020	Twilight	Stephenie ...	2021-01-03	Breaking Dawn	Novel

2) BORROWER TABLE

SAMPLE RECORDS OF BORROWER

```
/******DATA INSERT INTO TABLE******/  
  
/*INSERT VALUES*/  
  
INSERT INTO Borrower  
VALUES ('M001','Chamoth','Jayasekara','chamothJ@gmail.com','0701234567','20013656969v','COLOMBO','Kesbewa','2001-09-03','2021-09-03');  
  
INSERT INTO Borrower  
VALUES ('M002','Neranji','Sulakshika','nera@gmail.com','0701000000','20015000000v','COLOMBO','Kadawatha','2001-01-01','2021-01-01');  
  
INSERT INTO Borrower  
VALUES ('M003','Dhanuja','Sigera','dolly@gmail.com','0702000000','20013650001v','COLOMBO','Maharagama','2001-01-01','2021-01-01');  
  
INSERT INTO Borrower  
VALUES ('M004','Samadhi','Kaushalya','barrel@gmail.com','0703000000','200150000001','COLOMBO','Kaduwela','2001-01-01','2021-01-01');  
  
INSERT INTO Borrower  
VALUES ('M005','Dasith','Rupasingha','dizzy@gmail.com','0704000000','20013650002v','COLOMBO','Kesbewa','2001-01-01','2021-01-01');  
  
INSERT INTO Borrower  
VALUES ('M006','Irusha','Samarawikrama','thomas@gmail.com','0705000000','20013650003v','COLOMBO','Maharagama','2000-01-01','2020-01-01');  
  
INSERT INTO Borrower  
VALUES ('M007','N','J','god@gmail.com','0706000000','20013650004v','GALLE','Fort','2000-01-01','2020-01-01');  
  
INSERT INTO Borrower  
VALUES ('M008','Manoja','Perera','hi@gmail.com','0707000000','20015000002v','','Negombo','2000-01-01','2020-01-01');  
  
INSERT INTO Borrower  
VALUES ('M009','Mendika','Gurusinha','sidecut@gmail.com','0708000000','20015000003v','GALLE','','2000-01-01','2020-01-01');  
  
INSERT INTO Borrower  
VALUES ('M010','Pasindu','Galtikka','fishbun@gmail.com','0709000000','20015000004v','GALLE','','2000-01-01','2020-01-01');  
  
INSERT INTO Borrower  
VALUES ('M011','DB','Epawela','db@gmail.com','0700000000','20013650005v','GALLE','Fort','2002-01-01','2022-01-01');
```

- These are the sample data which has been inserted into the table 20 records in which only 15 have been inserted.
- Execution results

Results										
	M_ID	M_Fname	M_Lname	M_Email	M_Phone	M_NIC	M_District	M_City	M_Date	M_Expiry
1	M001	Chamoth	Jayasekara	chamothJ@gmail.com	701234567	20013656969v	COLOMBO	Kesbewa	2001-09-03	2021-09-03
2	M002	Neranji	Sulakshika	nera@gmail.com	701000000	20015000000v	COLOMBO	Kadawatha	2001-01-01	2021-01-01
3	M003	Dhanuja	Sigera	dolly@gmail.com	702000000	20013650001v	COLOMBO	Maharagama	2001-01-01	2021-01-01
4	M004	Samadhi	Kaushalya	barrel@gmail.com	703000000	200150000001	COLOMBO	Kaduwela	2001-01-01	2021-01-01
5	M005	Dasith	Rupasingha	dizzy@gmail.com	704000000	20013650002v	COLOMBO	Kesbewa	2001-01-01	2021-01-01
6	M006	Irusha	Samarawikrama	thomas@gmail.com	705000000	20013650003v	COLOMBO	Maharagama	2000-01-01	2020-01-01
7	M007	N	J	god@gmail.com	706000000	20013650004v	GALLE	Fort	2000-01-01	2020-01-01
8	M008	Manoja	Perera	hi@gmail.com	707000000	20015000002v		Negombo	2000-01-01	2020-01-01
9	M009	Mendika	Gurusinha	sidecut@gmail.com	708000000	20015000003v	GALLE		2000-01-01	2020-01-01
10	M010	Pasindu	Galtikka	fishbun@gmail.com	709000000	20015000004v	GALLE		2000-01-01	2020-01-01
11	M011	DB	Epawela	db@gmail.com	700000000	20013650005v	GALLE	Fort	2002-01-01	2022-01-01
12	M012	Mynameis	Library	pushthakaalaya@gma...	700000001	20015000056v	JAFFNA	Pedro	2002-01-01	2022-01-01
13	M013	Poth	Books	ilovebooks@gmail.c...	700000002	20015000043v	JAFFNA	Bay	2002-01-01	2022-01-01
14	M014	Harry	Potha	abarakadabra@gmail....	700000003	20013650045v	JAFFNA	Pedro	2002-01-01	2022-01-01
15	M015	Sharuk	Bhan	india@gmail.com	700000004	20013650032v	GALLE	Negombo	2002-01-01	2022-01-01

Query executed successfully.

3) COPY TABLE

1. Insert sample data

```
/*INSERTED*/
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B001', 'A01', 1050.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B002', 'B01', 1000.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B003', 'C01', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B003', 'C02', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B003', 'C03', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B004', 'D01', 1750.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B005', 'E01', 1975.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B005', 'E02', 1975.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B005', 'E03', 1975.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B006', 'F01', 1350.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B006', 'F02', 1350.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B007', 'G01', 1000.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B007', 'G02', 1000.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B007', 'G03', 1000.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B007', 'G04', 1000.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B008', 'H01', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B008', 'H02', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B009', 'I01', 1350.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B010', 'J01', 1750.00);

100 % ▾
Messages
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)

100 % ▾
Query executed successfully. | DESKTOP-VHPDJKD (15.0 RTM) | DESKTOP-VHF
Ln 59 Col 1 Ch 1 INS
```

SQL_Constraints (Copy).sql - DESKTOP-VHPDJKD.LibraryManagementSystem (DESKTOP-VHPDJKD\Neranji Sulakshika (58))

File Edit View Query Project Tools Window Help

LibraryManagementSystem

Object Explorer

Tables

- System Tables
- FileTables
- External Tables
- Graph Tables
- dbo.Book
- dbo.Borrower
- dbo.Copy
- Columns
- Keys
- Triggers
- Indexes
- Statistics
- dbo.Issuelnfo
- dbo.Publisher

SQL_Constraints (Co...nji Sulakshika (58))

```
SELECT * FROM Copy;
```

100 % ▾

Results Messages

	Copy_ISBN	CopyNo	Price
1	B001	A01	1115
2	B002	B01	1000
3	B003	C01	1500
4	B003	C02	1500
5	B003	C03	1500
6	B004	D01	1750
7	B005	E01	1950
8	B005	E02	1975
9	B005	E03	1975
10	B006	F02	1350
11	B007	G01	1000
12	B007	G02	1000
13	B007	G03	1000
14	B007	G04	1000
15	B008	H01	1500
16	B008	H02	1500
17	B009	I01	1350
18	B010	J01	1750
19	B011	K01	1975
20	B012	L01	1050
21	B013	M01	1100
22	B014	N01	1500
23	B014	N02	1500
24	B014	N03	1500
25	B015	O01	1750
26	B016	P01	1550
27	B016	P02	1550
28	B016	P03	1550
29	B017	Q01	1350
30	B017	Q02	1350
31	B018	R01	1490
32	B018	R02	1490
33	B018	R03	1490
34	B018	R04	1490
35	B019	S01	1630
36	B019	S02	1630
37	B020	T01	1350

Ready Ln 61

ElementSystem (DESKTOP-VHPDJKD\Neranji Sulakshika (57))* - Microsoft SQL Server Management Studio

SQL_Constraints (Co...nji Sulakshika (57))

```
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B011', 'K01', 1975.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B012', 'L01', 1050.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B013', 'M01', 1100.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B014', 'N01', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B014', 'N02', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B014', 'N03', 1500.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B015', 'O01', 1750.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B016', 'P01', 1550.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B016', 'P02', 1550.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B016', 'P03', 1550.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B017', 'Q01', 1350.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B017', 'Q02', 1350.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B018', 'R01', 1490.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B018', 'R02', 1490.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B018', 'R03', 1490.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B019', 'S01', 1630.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B019', 'S02', 1630.00);
Insert into Copy(Copy_ISBN, CopyNo, Price) VALUES ('B020', 'T01', 1350.00);

SELECT * FROM Copy;
```

100 % ▾

Messages

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2021-01-01T16:23:53.3662261+05:30

100 % ▾

Query executed successfully. | DESKTOP-VHPDJKD (15.0 RTM) | DESKTOP-VHF

SQL_Constraints (Copy).sql - DESKTOP-VHPDJKD.LibraryManagementSystem (DESKTOP-VHPDJKD\Neranji Sulakshika (58))

File Edit View Query Project Tools Window Help

LibraryManagementSystem

Object Explorer

Tables

- System Tables
- FileTables
- External Tables
- Graph Tables
- dbo.Book
- dbo.Borrower
- dbo.Copy
- Columns
- Keys
- Triggers
- Indexes
- Statistics
- dbo.Issuelnfo
- dbo.Publisher

SQL_Constraints (Co...nji Sulakshika (58))

```
SELECT * FROM Copy;
```

100 % ▾

Results Messages

	Copy_ISBN	CopyNo	Price
16	B008	H02	1500
17	B009	I01	1350
18	B010	J01	1750
19	B011	K01	1975
20	B012	L01	1050
21	B013	M01	1100
22	B014	N01	1500
23	B014	N02	1500
24	B014	N03	1500
25	B015	O01	1750
26	B016	P01	1550
27	B016	P02	1550
28	B016	P03	1550
29	B017	Q01	1350
30	B017	Q02	1350
31	B018	R01	1490
32	B018	R02	1490
33	B018	R03	1490
34	B018	R04	1490
35	B019	S01	1630
36	B019	S02	1630
37	B020	T01	1350

Ready Ln 61

4) ISSUE-INFO TABLE

1. Insert sample data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like IssueInfo, Copy, Borrower, and Book, along with their columns, keys, and constraints. The central pane displays a T-SQL script for inserting sample data into the IssueInfo table. The right pane shows the results of the executed query, displaying 12 rows of sample data.

```
/*
*****INSERT SAMPLE DATA*****
*/

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B001', 'A01', 'M001', '10.00', '2020.12.18', '2021.01.01');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B002', 'B01', 'M002', '10.00', '2020.12.18', '2021.01.01');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B003', 'C01', 'M003', '0.00', '2020.12.22', '2021.01.05');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B008', 'H01', 'M006', '', '2020.12.23', '2021.01.06');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B006', 'F02', 'M014', '10.00', '2020.12.23', '2021.01.06');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B010', 'J01', 'M019', '', '2020.12.24', '2021.01.07');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B004', 'D01', 'M010', '', '2020.12.25', '2021.01.08');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B005', 'E01', 'M017', '', '2020.12.27', '2021.01.10');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B007', 'G02', 'M020', '', '2020.12.27', '2021.01.10');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B009', 'I01', 'M004', '', '2020.12.31', '2021.01.14');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B011', 'K01', 'M008', '', '2021.01.01', '2021.01.15');

INSERT INTO IssueInfo (B_ISBN, CopyNo, Member_ID, Fine, IssueDate, ReturnDate)
VALUES ('B013', 'M01', 'M005', '', '2021.01.01', '2021.01.15');

SELECT * FROM IssueInfo;
```

	IssueNo	B_ISBN	CopyNo	Member_ID	Fine	IssueDate	ReturnDate
1	1	B001	A01	M001	10	2020-12-18	2021-01-01
2	2	B002	B01	M002	10	2020-12-18	2021-01-01
3	3	B003	C01	M003	0	2020-12-22	2021-01-05
4	4	B008	H01	M006	0	2020-12-23	2021-01-06
5	5	B006	F02	M014	10	2020-12-23	2021-01-06
6	6	B010	J01	M019	0	2020-12-24	2021-01-07
7	7	B004	D01	M010	0	2020-12-25	2021-01-08
8	8	B005	E01	M017	0	2020-12-27	2021-01-10
9	9	B007	G02	M020	0	2020-12-27	2021-01-10
10	10	B009	I01	M004	0	2020-12-31	2021-01-14
11	11	B011	K01	M008	0	2021-01-01	2021-01-15
12	12	B013	M01	M005	0	2021-01-01	2021-01-15

5) STAFF TABLE

Insert table staff

Started inserting data(s) into the Staff Table,

```
INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Dasitha','Samarasinghe','0772899112','dsamare21@gmail.com','Bookkeeper','2002-01-19');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Shevon','Marasinghe','0772431132','shemara67@gmail.com','Cleaner','2005-03-15');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Chamodh','Jayasekara','0772569112','chamo41@gmail.com','Bookkeeper','2005-08-23');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Nethmi','Sulakshika','0772899432','N5sulakshika23@gmail.com','Cleaner','2006-07-26');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Dhanuja','Sigera','0776783242','djvandiesl@yahoo.com','Ledger Holder','2008-12-12');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Kevin','Koththigoda','0768392313','kevinkoththi90@gmail.com','Bookkeeper','2009-01-02');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Dhanuska','Samarasinghe','0772892312','danusamare12@gmail.com','Bookkeeper','2009-09-18');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Pathme','Sirisenage','0777829112','pathme23@gmail.com','Guard','2009-09-12');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Romesh','Udana','0722499112','Cleaner','2012-03-12');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Rohitha','Gunaratne','0732456722','rohitagune32@yahoo.com','Guard','2014-02-23');

INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Vindya','Ravindi','0732499112','vini45@yahoo.com','Admin','2013-02-24');
```

After inserted data into the table, this is result,

FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date	
Dasitha	Samarasinghe	772899112	dsamare21@gmail.com	1	Bookkeeper	2002-01-19	
Shevon	Marasinghe	772431132	shemara67@gmail.com	2	Cleaner	2005-03-15	
Chamodh	Jayasekara	772569112	chamo41@gmail.com	3	Bookkeeper	2005-08-23	
Nethmi	Sulakshika	772899432	N5sulakshika23@gmail.com	4	Cleaner	2006-07-26	
Dhanuja	Sigera	776783242	djvandiesl@yahoo.com	5	Ledger Holder	2008-12-12	
Kevin	Koththigoda	768392313	kevinkoththi90@gmail.com	6	Bookkeeper	2009-01-02	
Dhanuska	Samarasinghe	772892312	danusamare12@gmail.com	7	Bookkeeper	2009-09-18	
Pathme	Sirisenage	777829112	pathme23@gmail.com	8	Guard	2009-09-12	
Romesh	Udana	722499112	NULL	9	Cleaner	2012-03-12	
0	Rohitha	Gunaratne	732456722	rohitagune32@yahoo.com	10	Guard	2014-02-23

6) PUBLISHER TABLE

After creating the publisher table, we need to insert data into the publisher table.

```
VALUES ('1003', 'Penguin books', 'B004', 'United States', ' ', '2000/7/20', '0118545871', 'penguinbooks@ymail.com');

INSERT INTO Publisher
VALUES ('1004', 'Scolastic press', 'B005', 'United Kingdom', 'Texas', '2003/1/26', '0715224871', 'scolasticpress123@gmail.com');

INSERT INTO Publisher
VALUES ('1005', 'VI Publishers', 'B006', 'Australia', 'Perth', '2005/5/14', '0524745871', 'vipublishers@gmail.com');

INSERT INTO Publisher
VALUES ('1006', 'Aspen Publishers', 'B007', 'United States', ' ', '2003/1/26', '0715445871', 'aspenpublishers@hotmail.com');

INSERT INTO Publisher
VALUES ('1007', 'Blue Whale Press', 'B008', 'United States', 'New York', '2007/2/15', '0112857871', 'bluewhales@gmail.com');

INSERT INTO Publisher
VALUES ('1008', 'Scribner press', 'B009', 'United States', 'Chicago', '1997/1/16', '0047845871', 'scibnepress@ymail.com');

INSERT INTO Publisher
VALUES ('1009', 'Archid constable and company', 'B010', 'United Kingdom', 'Bristol', '1897/6/26', '0769874471', 'archidconstablecompany@gmail.com');

INSERT INTO Publisher
VALUES ('1010', 'Pantheon house', 'B011', 'United Kingdom', ' ', '2000/3/07', '0114785371', 'pantheonhouse123@hotmail.com');

INSERT INTO Publisher
VALUES ('1011', 'DC Comics', 'B012', ' ', ' ', ' ', '022489594', 'dccomics@yahoo.com');

% - Messages

(1 row affected)
0 % - 
Query executed successfully. LAPTOP-F1BNF1S2SOLEXPRESS LAPTOP-F1BNF1S2samad Library 000
```

After executing the values, below are the results of the entered values to the database.

```
SELECT * FROM Publisher; /* DISPLAYS THE PUBLISHER TABLE WITH THE RECORDS INSERTED */

100 % - 
Results Messages



| Pub_ID | Pub_Name                     | Published_BookID | Pub_Country    | Pub_City | Pub_Date   | Pub_Contact | Pub_Email                        |
|--------|------------------------------|------------------|----------------|----------|------------|-------------|----------------------------------|
| 1      | Bloomsbury Publishing        | B001             | United Kingdom | London   | 1997-06-26 | 0112445871  | bloomsbury.publishing@yahoo.com  |
| 2      | United Book Distributors     | B002             | Australia      | Sydney   | 1998-08-15 | 0225345871  | uniteddistributors000@gmail.com  |
| 3      | Macmillans                   | B003             | United States  | New York | 1999-05-02 | 0557484871  | macmillans123@gmail.com          |
| 4      | Penguin books                | B004             | United States  |          | 2000-07-20 | 0118545871  | penguinbooks@ymail.com           |
| 5      | Scolastic press              | B005             | United Kingdom | Texas    | 2003-01-26 | 0715224871  | scolasticpress123@gmail.com      |
| 6      | VI Publishers                | B006             | Australia      | Perth    | 2005-05-14 | 0524745871  | vipublishers@gmail.com           |
| 7      | Aspen Publishers             | B007             | United States  |          | 2003-01-26 | 0715445871  | aspenpublishers@hotmail.com      |
| 8      | Blue Whale Press             | B008             | United States  | New York | 2007-02-15 | 0112857871  | bluewhales@gmail.com             |
| 9      | Scribner press               | B009             | United States  | Chicago  | 1997-01-16 | 0047845871  | scibnepress@ymail.com            |
| 10     | Archid constable and company | B010             | United Kingdom | Bristol  | 1897-06-26 | 0769874471  | archidconstablecompany@gmail.com |
| 11     | Pantheon house               | B011             | United Kingdom |          | 2000-03-07 | 0114785371  | pantheonhouse123@hotmail.com     |
| 12     | DC Comics                    | B012             |                |          | 1900-01-01 | 022489594   | dccomics@yahoo.com               |
| 13     | Simon & Shuster              | B013             | United Kingdom | Cardiff  | 2014-04-15 | 0112478871  | simon&shutter2@gmail.com         |


```

7) LIBRARY TABLE

```
SELECT * FROM library;
insert into library values ('003' , 'Branch','NSBM','school','homagama', '31/45' , '03')
insert into library values ('002' , 'Branch','IBT','school','colombo', '11/45' , '05')
insert into library values ('001' , 'main','nex','public','colombo', '12/45' , '63')
insert into library values ('004' , 'branch','nex','public','kandy', '15/45' , '33')
insert into library values ('005' , 'main','NSBM','school','colombo', '18/45' , '43')
insert into library values ('006' , 'branch','norton','private','colombo', '10/45' , '72')
insert into library values ('007' , 'branch','leeds','school','Galle', '60/45' , '92')
insert into library values ('008' , 'main','weed','private','colombo', '05/45' , '70')
insert into library values ('009' , 'branch','learn it','public','Thalawathugoda', '54/45' , '32')
insert into library values ('010' , 'branch','oxford','public','Thalawathugoda', '04/45' , '27')
```

Results							
libcode	LibraryType	BranchName	BranchType	City	LibNumber	CopyNumber	
1	001	main	nex	public	colombo	12/45	63
2	002	Branch	IBT	school	colombo	11/45	05
3	003	Branch	NSBM	school	homagama	31/45	03
4	004	branch	nex	public	kandy	15/45	33
5	005	main	NSBM	school	colombo	18/45	43
6	006	branch	norton	private	colombo	10/45	72
7	007	branch	leeds	school	Galle	60/45	92
8	008	main	weed	private	colombo	05/45	70
9	009	branch	learn it	public	Thalawathugoda	54/45	32
10	010	branch	oxford	public	Thalawathugoda	04/45	27

This the sample data for the above created table (library)

3. SECTION 03

❖ TRIGGERS

1) BOOK TABLE

➤ INSERT TRIGGERS

This shows the creation of the trigger.

The screenshot shows a SQL query editor window with the following content:

```
/*****TRIGGERS*****  
CREATE TRIGGER Book_Insert  
ON Book  
AFTER INSERT  
AS  
    SELECT ISBN, B_Name, B_Author, B_Category  
    FROM Book;  
    INSERT INTO Book(ISBN, B_Name, B_Author, B_Category)  
    VALUES('B015', 'Goosebumps', 'R.L.Stine', 'Horro fiction');  
  
    SELECT * FROM Book;  
  
DROP TRIGGER IF EXISTS Book_Insert;
```

Below the code, there is a progress bar at 0.00% and a message area stating "Commands completed successfully." with a timestamp: "Completion time: 2021-01-03T18:34:25.4408091+05:30".

➤ UPDATE TRIGGERS

```
CREATE TRIGGER BookDate_Update
ON Book
AFTER UPDATE
AS
BEGIN
    set nocount on;
    UPDATE book set B_PublishedDate = getdate()
    FROM Book b
    INNER JOIN inserted i on b.ISBN = i.ISBN
    AND i.B_Name = 'Approved'
END
GO

UPDATE Book set B_Name = 'Approved'
WHERE ISBN = 'B015'
GO

SELECT * FROM Book
```

0 %

! Results Messages

ISBN	B_Name	B_Author	B_PublishedDate	B_Title	B_Category
B001	Harry Potter	J. K. Rowling	2021-01-03	and the Philosophers Stone	Novel
B002	Harry Potter	J. K. Rowling	2021-01-03	and the Deathly Hallows	Novel
B003	Harry Potter	J. K. Rowling	2021-01-03	and the Cursed Child	Novel
B004	Harry Potter	J. K. Rowling	2021-01-03	and the Goblet of Fire	Novel
B005	Harry Potter	J. K. Rowling	2021-01-03	and the Order of the Phoenix	Novel
B006	Harry Potter	J. K. Rowling	2021-01-03	and the Chamber of Secrets	Novel
B007	Harry Potter	J. K. Rowling	2021-01-03	and the Prisoner of Azkaban	Novel
B008	Cinderella	Daisy Fisher	2021-01-03		Fairy tale
B009	Dracula	Bram Stoker	2021-01-03		Novel
) B010	House of...	Mark Z. Da...	2021-01-03		Novel
1 B011	Wonder ...	Leigh Bard...	2021-01-03	Warbringer	Fiction
2 B012	The Chro...	C.S.Lewis	2021-01-03	Narnia	Novel
3 B013	Twilight	Stephenie ...	2021-01-03	New moon	Novel
4 B014	A Promise...	Barack Ob...	2021-01-03		Autobiogr...
5 B015	Approved	R. L. Stine	2021-01-03	The Haunted Mask	Horror fict...
6 B016	Goosebumps	R.L. Stine	2021-01-03	Monsters in Dead House	Horror fict...

- This trigger is to update an existing value in the table.

➤ DELETING TRIGGER

```
*****TRIGGERS*****
```

```
CREATE TRIGGER Book_Insert
ON Book
FOR INSERT
AS
    SELECT ISBN, B_Name, B_Author, B_Category
    FROM Book;
    INSERT INTO Book(ISBN, B_Name, B_Author, B_Category)
    VALUES('B015', 'Goosebumps', 'R.L.Stine', 'Horro fiction');

    DROP TRIGGER IF EXISTS Book_Insert;
```

100 %

! Messages

Commands completed successfully.

Completion time: 2021-01-03T18:16:54.0485160+05:30

2) BORROWER TABLE

TRIGGERS FOR BORROWER

INSERT TRIGGER

```
/******TRIGGERS*****/
/*SQL TRIGGERS FOR MEMBER INSERT*/

CREATE TRIGGER member_insert
ON Borrower
AFTER INSERT
AS
BEGIN
DECLARE @m_id varchar(4)
SELECT @m_id = M_ID from Inserted
INSERT INTO new_member
VALUES (@m_id, 'New Member with Membership ID = ' + CAST (@m_id as varchar (4)) + 'has joined to the library system as at ' + CAST (GETDATE() as varchar(100)))
SELECT * FROM Borrower;
SELECT * FROM new_member;
END
```

- This trigger is created to trigger an event of which when a new member is being entered into the database the borrower table will be shown with the new entry along with another table showing that this person with the related M_ID has been entered into the database.
- Executed result

The screenshot shows the SSMS interface with the following details:

- Query Editor:** Contains the SQL code for inserting a new member into the Borrower table:

```
INSERT INTO Borrower
VALUES ('M007', 'N', 'J', 'god@gmail.com', '0706000000', '20013650004v', 'GALLE', 'Fort', '2000-01-01', '2020-01-01');
```
- Results Tab:** Displays the updated data in the Borrower table. A red box highlights the newly inserted row (M007).

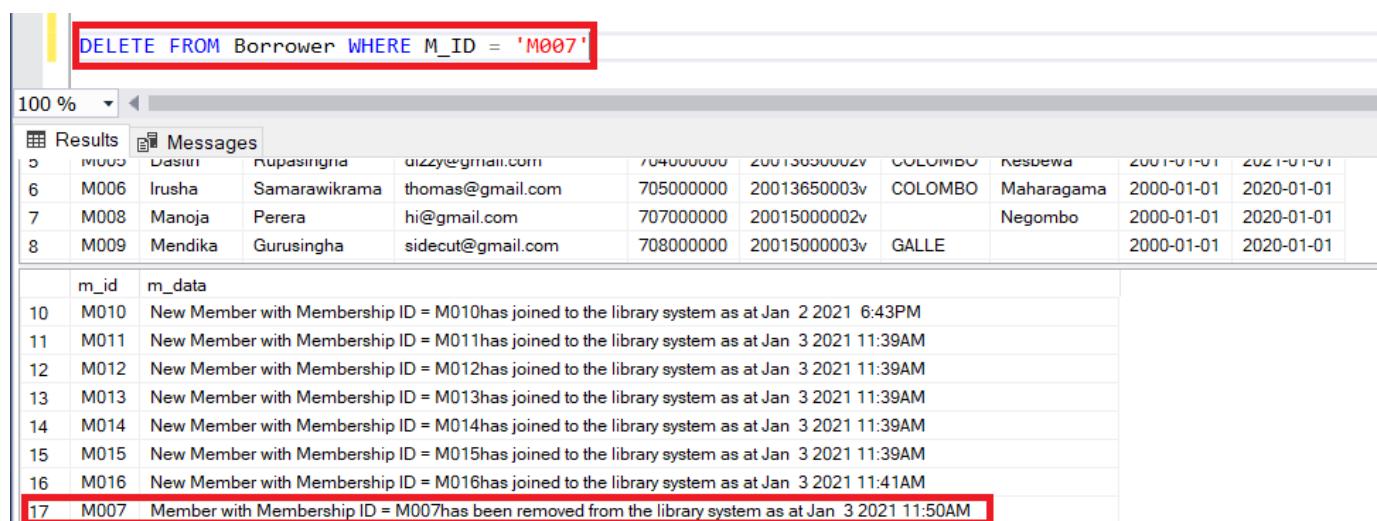
m_id	l_name	f_name	email	tel	reg_no	branch	residence	join_date	exp_date
M006	Irusha	Samarawikrama	thomas@gmail.com	705000000	20013650003v	COLOMBO	Maharagama	2000-01-01	2020-01-01
M007	N	J	god@gmail.com	706000000	20013650004v	GALLE	Fort	2000-01-01	2020-01-01
M008	Manoja	Perera	hi@gmail.com	707000000	20015000002v		Negombo	2000-01-01	2020-01-01
- Messages Tab:** Shows log messages for the trigger execution, with a red box highlighting the message for the new member insertion.

m_id	m_data
M011	New Member with Membership ID = M011has joined to the library system as at Jan 3 2021 11:39AM
M012	New Member with Membership ID = M012has joined to the library system as at Jan 3 2021 11:39AM
M013	New Member with Membership ID = M013has joined to the library system as at Jan 3 2021 11:39AM
M014	New Member with Membership ID = M014has joined to the library system as at Jan 3 2021 11:39AM
M015	New Member with Membership ID = M015has joined to the library system as at Jan 3 2021 11:39AM
M016	New Member with Membership ID = M016has joined to the library system as at Jan 3 2021 11:41AM
M007	Member with Membership ID = M007has been removed from the library system as at Jan 3 2021 11:50AM
M007	New Member with Membership ID = M007has joined to the library system as at Jan 3 2021 11:59AM
- Status Bar:** Shows "Query executed successfully." and the connection information "(LocalDB)\MSSQLLocalDB".

DELETE TRIGGER

```
/*SQL TRIGGERS FOR MEMBER DELETE*/  
  
CREATE TRIGGER member_delete  
ON Borrower  
AFTER DELETE  
AS  
BEGIN  
DECLARE @m_id varchar(4)  
SELECT @m_id = M_ID from deleted  
INSERT INTO new_member  
VALUES (@m_id, 'Member with Membership ID = ' + CAST (@m_id as varchar (4)) + 'has been removed from the library system as at ' + CAST (GETDATE() as varchar(100)))  
SELECT * FROM Borrower;  
SELECT * FROM new_member;  
END
```

- This trigger is created to trigger an event of which when a member is being deleted from the database the borrower table will be shown along with another table showing that this person with the related M_ID has been removed from the database. As shown below:
- Executed Result



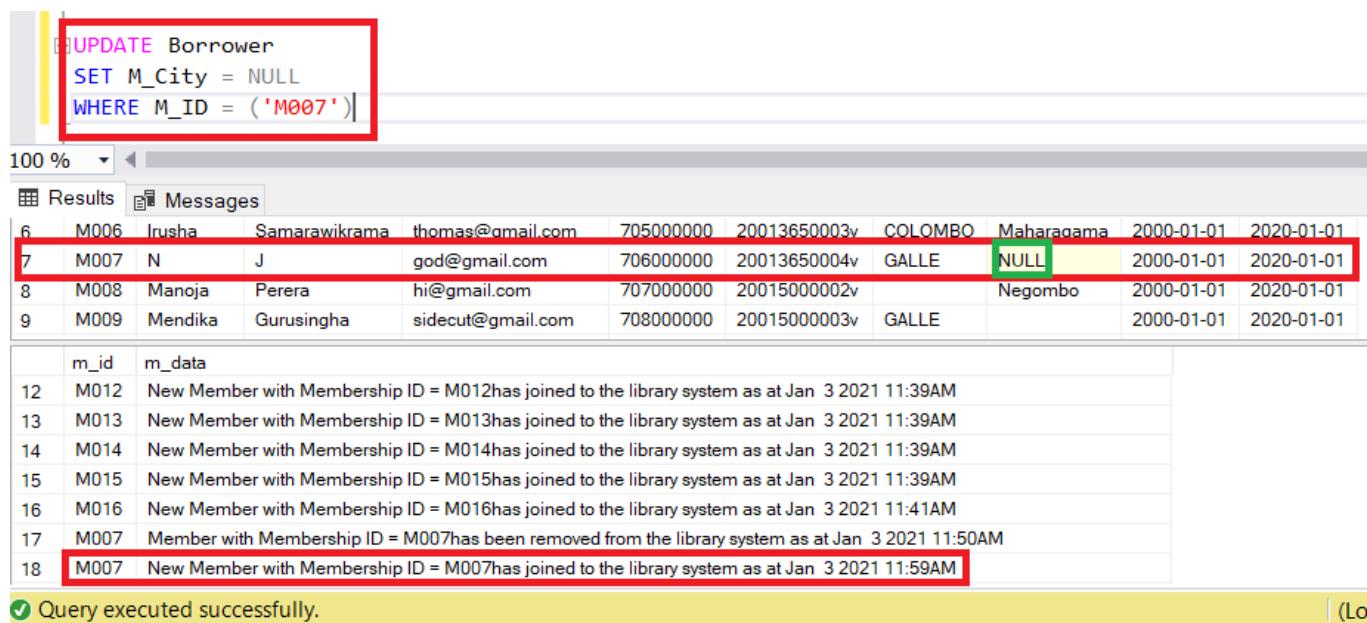
The screenshot shows the SQL Server Management Studio interface. In the top-left pane, a query window contains the command: `DELETE FROM Borrower WHERE M_ID = 'M007'`. In the bottom-right pane, there are two result sets. The first result set is titled 'Results' and shows the 'Borrower' table with 8 rows of data. The second result set is titled 'Messages' and shows the 'new_member' table with 17 rows of data, each containing a message about a member joining or being removed from the library system.

	m_id	m_data
10	M010	New Member with Membership ID = M010has joined to the library system as at Jan 2 2021 6:43PM
11	M011	New Member with Membership ID = M011has joined to the library system as at Jan 3 2021 11:39AM
12	M012	New Member with Membership ID = M012has joined to the library system as at Jan 3 2021 11:39AM
13	M013	New Member with Membership ID = M013has joined to the library system as at Jan 3 2021 11:39AM
14	M014	New Member with Membership ID = M014has joined to the library system as at Jan 3 2021 11:39AM
15	M015	New Member with Membership ID = M015has joined to the library system as at Jan 3 2021 11:39AM
16	M016	New Member with Membership ID = M016has joined to the library system as at Jan 3 2021 11:41AM
17	M007	Member with Membership ID = M007has been removed from the library system as at Jan 3 2021 11:50AM

UPDATE TRIGGER

```
/*SQL TRIGGERS FOR MEMBER UPDATE*/  
  
CREATE TRIGGER member_update  
ON Borrower  
AFTER UPDATE  
AS  
BEGIN  
DECLARE @m_id varchar(4)  
SELECT @_id = M_ID from inserted  
INSERT INTO new_member  
VALUES (@_id, 'Member with Membership ID = ' + CAST (@_id as varchar (4)) + ' has been updated from the library system as at ' + CAST (GETDATE() as varchar(100)))  
SELECT * FROM Borrower;  
SELECT * FROM new_member;  
END
```

- This trigger is created to trigger an event of which when a member's details is being updated from the database the borrower table will be shown with the newly updated entry along with another table showing that this person with the related M_ID's details had been updated from the database. As shown below:
- Executed result



The screenshot shows the MySQL Workbench interface. A red box highlights the SQL query:

```
UPDATE Borrower  
SET M_City = NULL  
WHERE M_ID = ('M007')
```

The results pane shows the Borrower table with data rows 6 through 9. Row 7 (M007) has its M_City column set to NULL. The log pane at the bottom shows the following entries:

m_id	m_data
12	M012 New Member with Membership ID = M012has joined to the library system as at Jan 3 2021 11:39AM
13	M013 New Member with Membership ID = M013has joined to the library system as at Jan 3 2021 11:39AM
14	M014 New Member with Membership ID = M014has joined to the library system as at Jan 3 2021 11:39AM
15	M015 New Member with Membership ID = M015has joined to the library system as at Jan 3 2021 11:39AM
16	M016 New Member with Membership ID = M016has joined to the library system as at Jan 3 2021 11:41AM
17	M007 Member with Membership ID = M007has been removed from the library system as at Jan 3 2021 11:50AM
18	M007 New Member with Membership ID = M007has joined to the library system as at Jan 3 2021 11:59AM

A yellow bar at the bottom indicates "Query executed successfully."

3) COPY TABLE

1. After Update Trigger

- Create and execute the ‘AfterUpdateTrigger’.

The screenshot shows two instances of Microsoft SQL Server Management Studio (SSMS) running side-by-side. Both instances are connected to the same database, 'LibraryManagementSystem'.

Left Instance (Trigger Creation):

- Object Explorer:** Shows the database structure, including tables like 'Copy', 'Book', 'Publisher', and 'Author'. A trigger named 'AfterUPDATETrigger' is listed under the 'Triggers' node for the 'Copy' table.
- SQL Editor:** Displays the T-SQL code for creating the trigger:


```

      /* AFTER UPDATE */
CREATE TRIGGER AfterUPDATETrigger ON [Copy]
FOR UPDATE
AS DECLARE @CopyNo varchar(10),
          @Price float;

SELECT @CopyNo = ins.CopyNo FROM INSERTED ins;
SELECT @Price = ins.Price FROM INSERTED ins;

/*After..*/
SELECT * FROM Copy WHERE CopyNo = 'E01';

UPDATE [Copy]
SET [Price] = 1950.00
WHERE CopyNo = 'E01';

SELECT * FROM Copy WHERE Copy_ISBN= 'B005';
      
```
- Messages:** Shows the message "Commands completed successfully." and the completion time: 2021-01-01T16:24:49.8902405+05:30.
- Status Bar:** Shows "Query executed successfully.", the connection name "DESKTOP-VHPDJKD", and the current position: Ln 100 Col 1 Ch 1.

Right Instance (Trigger Execution):

- Object Explorer:** Shows the same database structure.
- SQL Editor:** Displays the same T-SQL code as the left instance.
- Results:** Shows the output of the SELECT query:

Copy_ISBN	CopyNo	Price
B005	E01	1975
- Messages:** Shows the message "Query executed successfully." and the completion time: 2021-01-01T16:24:49.8902405+05:30.
- Status Bar:** Shows "Query executed successfully.", the connection name "DESKTOP-VHPDJKD", and the current position: Ln 78 Col 1 Ch 1.

Bottom Left Instance (Trigger Execution with Data Insert):

- Object Explorer:** Shows the database structure.
- SQL Editor:** Displays the same T-SQL code as the other instances.
- Results:** Shows the output of the SELECT query:

Copy_ISBN	CopyNo	Price
B005	E01	1950
- Messages:** Shows the message "(1 row affected)" and the completion time: 2021-01-01T16:25:25.8584810+05:30.
- Status Bar:** Shows "Query executed successfully.", the connection name "DESKTOP-VHPDJKD", and the current position: Ln 81 Col 25 Ch 22.

Bottom Right Instance (Trigger Execution with Data Update):

- Object Explorer:** Shows the database structure.
- SQL Editor:** Displays the same T-SQL code as the other instances.
- Results:** Shows the output of the SELECT query:

Copy_ISBN	CopyNo	Price
B005	E02	1975
B005	E03	1975
- Messages:** Shows the message "Query executed successfully." and the completion time: 2021-01-01T16:25:25.8584810+05:30.
- Status Bar:** Shows "Query executed successfully.", the connection name "DESKTOP-VHPDJKD", and the current position: Ln 84 Col 1 Ch 1.

1. After Delete Trigger

- Create and execute the ‘AfterUpdateTrigger’.

The screenshot shows the Object Explorer on the left with the LibraryManagementSystem database selected. In the center, the SQL Editor window displays the following T-SQL code:

```

/* AFTER DELETE */

CREATE TRIGGER AfterDELETETrigger on [Copy]
FOR DELETE
AS DECLARE @CopyNo varchar(10);

SELECT @CopyNo = del.CopyNo FROM DELETED del;

SELECT * FROM Copy WHERE Copy_ISBN= 'B006';

/*After..*/
DELETE FROM [Copy]
WHERE [CopyNo] = 'F01'

SELECT * FROM Copy WHERE Copy_ISBN= 'B006';

```

The Messages pane at the bottom right shows "Commands completed successfully." and the completion time: 2021-01-01T16:25:48.3527197+05:30. The status bar at the bottom indicates "Query executed successfully." and the session details: DESKTOP-VHPDJKD\Neranji Sulakshika (57)*.

The screenshot shows two side-by-side SQL Editor windows. Both windows contain the same T-SQL code as the previous screenshot, but they show different results in their Results panes.

Left Window Results:

	Copy_ISBN	CopyNo	Price
1	B006	F02	1350

Right Window Results:

	Copy_ISBN	CopyNo	Price
1	B006	F02	1350

Both windows show the message "Query executed successfully." in their status bars. The left window's status bar also shows "Ln 101 Col 5".

4) ISSUE-INFO TABLE

1. After Update Trigger

The screenshot shows the Object Explorer on the left with the 'dbo.IssueInfo' table selected. The 'Triggers' folder under 'dbo.IssueInfo' contains a single item: 'AfterUPDATETriggerIssueInfo'. The 'SQL Constraints' window on the right displays the trigger script:

```

/*
*****TRIGGERS*****
*/
/* AFTER UPDATE */

CREATE_TRIGGER AfterUPDATETriggerIssueInfo ON [IssueInfo]
FOR UPDATE
AS DECLARE @IssueNo int,
          @ReturnDate date;

SELECT @IssueNo = ins.IssueNo FROM INSERTED ins;
SELECT @ReturnDate = ins.ReturnDate FROM INSERTED ins;

/*Execute...*/
UPDATE [IssueInfo]
   SET [ReturnDate] = '2021.01.02'
 WHERE IssueNo = 1

SELECT * FROM IssueInfo;

```

The 'Messages' pane at the bottom shows the command completed successfully.

This After Update Trigger is created to update the return date

The screenshot shows the Object Explorer on the left with the 'dbo.IssueInfo' table selected. The 'Triggers' folder under 'dbo.IssueInfo' contains a single item: 'AfterUPDATETriggerIssueInfo'. The 'SQL Constraints' window on the right displays the trigger script, identical to the one in the previous screenshot.

The 'Results' pane at the bottom shows the data from the 'IssueInfo' table:

	IssueNo	B_ISBN	CopyNo	Member_ID	Fine	IssueDate	ReturnDate
1	1	B001	A01	M001	10	2020-12-18	2021-01-02
2	2	B002	B01	M002	10	2020-12-18	2021-01-01
3	4	B008	H01	M006	0	2020-12-23	2021-01-06
4	5	B006	F02	M014	10	2020-12-23	2021-01-06
5	6	B010	J01	M019	0	2020-12-24	2021-01-07
6	7	B004	D01	M010	0	2020-12-25	2021-01-08
7	8	B005	E01	M017	0	2020-12-27	2021-01-10
8	9	B007	G02	M020	0	2020-12-27	2021-01-10
9	10	B009	I01	M004	0	2020-12-31	2021-01-14
10	11	B011	K01	M008	0	2021-01-01	2021-01-15
11	12	B013	M01	M005	0	2021-01-01	2021-01-15

The 'Messages' pane at the bottom shows the command completed successfully.

After executing this, we can see 'IssueNo 1' related Return date updated to '2020.01.02'.

2. After Delete Trigger

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the Library_Management_System database, the IssueInfo table is selected. In the center pane, a new trigger named 'AfterDELETETriggerIssueInfo' is being created on the IssueInfo table for the DELETE event. The trigger's code is as follows:

```

/* AFTER DELETE */
CREATE TRIGGER AfterDELETETriggerIssueInfo ON [IssueInfo]
FOR DELETE
AS DECLARE @IssueNo int;

SELECT @IssueNo = del.IssueNo FROM DELETED del;

/*Execute...*/
DELETE FROM [IssueInfo]
WHERE [IssueNo] = 3

SELECT * FROM IssueInfo;

```

The 'Messages' pane at the bottom right shows the command completed successfully.

This After Delete Trigger is created to delete the 'IssueNo 3' related all data.

The screenshot shows the Microsoft SQL Server Management Studio interface. The same trigger creation steps are shown again. After executing the trigger, the results pane displays the data from the IssueInfo table. The row where IssueNo = 3 has been deleted is highlighted with a red border.

	IssueNo	B_ISBN	CopyNo	Member_ID	Fine	IssueDate	ReturnDate
1	1	B001	A01	M001	10	2020-12-18	2021-01-02
2	2	B002	B01	M002	10	2020-12-18	2021-01-01
3	4	B008	H01	M006	0	2020-12-23	2021-01-06
4	5	B006	F02	M014	10	2020-12-23	2021-01-06
5	6	B010	J01	M019	0	2020-12-24	2021-01-07
6	7	B004	D01	M010	0	2020-12-25	2021-01-08
7	8	B005	E01	M017	0	2020-12-27	2021-01-10
8	9	B007	G02	M020	0	2020-12-27	2021-01-10
9	10	B009	I01	M004	0	2020-12-31	2021-01-14
10	11	B011	K01	M008	0	2021-01-01	2021-01-15
11	12	B013	M01	M005	0	2021-01-01	2021-01-15

After execut this, we can see 'IssueNo 3' related data is removed successfully.

5) STAFF TABLE

After Insert Trigger

I created a trigger to insert the joined date with using GETDATE function as this will help to find who are the new members with their primary key, StaffID.

```
CREATE TRIGGER Date_Insert
ON StaffTable
AFTER INSERT
AS
BEGIN
DECLARE @id INT
SELECT @id = StaffID from inserted
INSERT INTO StaffTable VALUES (@id, CAST(GETDATE() as date))
SELECT * FROM newmembers
SELECT * FROM StaffTable
END
```

To find members who joined recently, I can use that execute by following command,

The screenshot shows the SQL Server Management Studio interface with two tabs: 'Results' and 'Messages'. The 'Results' tab displays two tables of data. The first table, titled 'joineddate', has columns 'Staff_Id' and 'joineddate', containing the following data:

	Staff_Id	joineddate
1	18	2021-01-02
2	15	2021-01-03
3	16	2021-01-03
4	19	2021-01-03

The second table, titled 'StaffTable', has columns 'FirstName', 'LastName', 'Contact', 'Email', 'StaffID', 'StaffRole', and 'Joined_Date', containing the following data:

	FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date
1	Dasitha	Samarasinghe	772899112	dsamare21@gmail.com	1	Bookkeeper	2002-01-19
2	Shevon	Marasinghe	772431132	shemara67@gmail.com	2	Cleaner	2005-03-15
3	Chamodh	Jayasekara	772569112	chamo41@gmail.com	3	Bookkeeper	2005-08-23
4	Nethmi	Sulakshika	772899432	N5sulakshika23@gma...	4	Cleaner	2006-07-26

A yellow status bar at the bottom of the results pane says 'Query executed successfully.'

After Update Trigger

```
CREATE TRIGGER updatestaff
ON StaffTable
AFTER UPDATE
AS
BEGIN
SELECT * FROM StaffTable
END
```

With that when every single update event like this,

The screenshot shows a SQL Server Management Studio (SSMS) interface. The top pane contains T-SQL code. The bottom pane has tabs for 'Results' and 'Messages'. The 'Results' tab displays a grid of data from the 'StaffTable'.

```
END

UPDATE StaffTable
SET StaffRole = 'Cleaner'
WHERE StaffID = 7 ;

CREATE TRIGGER deletestaff
ON StaffTable
AFTER DELETE
AS
BEGIN
SELECT * FROM StaffTable
END

DELETE FROM StaffTable where StaffID = 11

CREATE FUNCTION findjoineddate (@id int)
returns date
as
```

FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date
Dasitha	Samarasinghe	772899112	dsamare21@gmail.com	1	Bookkeeper	2002-01-19
Shevon	Marasinghe	772431132	shemara67@gmail.com	2	Cleaner	2005-03-15
Chamodh	Jayasekara	772569112	chamo41@gmail.com	3	Bookkeeper	2005-08-23
Nethmi	Sulakshika	772899432	N5sulakshika23@gmail.com	4	Cleaner	2006-07-26
Dhanuja	Sigera	776783242	djvandiesl@yahoo.com	5	Ledger Holder	2008-12-12
Kevin	Koththigoda	768392313	kevinkoththi90@gmail.com	6	Bookkeeper	2009-01-02
Dhanuska	Samarasinghe	772892312	danusamare12@gmail.com	7	Cleaner	2009-09-18
Pathme	Sirisenage	777829112	pathme23@gmail.com	8	Guard	2009-09-12
Romesh	Udama	722499112	NULL	9	Cleaner	2012-03-12
				10		2014-02-20

Query executed successfully

After Delete Trigger

```
CREATE TRIGGER deletestaff
ON StaffTable
AFTER DELETE
AS
BEGIN
SELECT * FROM StaffTable
END

DELETE FROM StaffTable where StaffID = 14
```

When delete the staff id = 14 automatically the table will shown up in execution window like this,

	FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date
1	Dasitha	Samarasinghe	772899112	dsamare21@gmail.com	1	Bookkeeper	2002-01-19
2	Shevon	Marasinghe	772431132	shemara67@gmail.com	2	Cleaner	2005-03-15
3	Chamodh	Jayasekara	772569112	chamo41@gmail.com	3	Bookkeeper	2005-08-23
4	Nethmi	Sulakshika	772899432	N5sulakshika23@gmail.com	4	Cleaner	2006-07-26
5	Dhanuja	Sigera	776783242	djvandiesl@yahoo.com	5	Ledger Holder	2008-12-12
6	Kevin	Koththigoda	768392313	kevinkoththi90@gmail.com	6	Bookkeeper	2009-01-02
7	Dhanuska	Samarasinghe	772892312	danusamare12@gmail.com	7	Cleaner	2009-09-18
8	Pathme	Sirisenage	777829112	pathme23@gmail.com	8	Guard	2009-09-12
9	Romesh	Udana	722499112	NULL	9	Cleaner	2012-03-12
10	Rohitha	Gunaratne	732456722	rohitagune32@yahoo.com	10	Guard	2014-02-23
11	Vindya	Ravindi	732499112	vini45@yahoo.com	11	Admin	2013-02-24
12	Abdul	Rashid	732349112	rashid321@yahoo.com	15	Bookledger	2021-01-03
13	Vindi	Narandhi	747499154	never23@yahoo.com	16	Bookledger	2018-05-23
14	Irusha	Samarawickr...	733449112	iruu321@yahoo.com	19	Bookledger	2021-01-03

6) PUBLISHER TABLE

- Creation of a trigger

The screenshot shows a SQL query window with the following code:

```
//** CAN BE USED TO CREATE DEFAULT VALUE OF COLOMBO**/ 

CREATE TRIGGER Pub_Insert /* TRIGGER USED TO INSERT VALUES INTO THE PUBLISHER TABLE */ 
ON Publisher 
FOR INSERT 
AS 
    SELECT Pub_ID , Pub_Name , Pub_Country 
    FROM Publisher
```

Below the code, the message "Commands completed successfully." is displayed, along with the completion time: 2021-01-03T17:25:16.2945965+05:30.

The above shows how to create a trigger.

- Data insertion and the result after executing the trigger.
- Here there is a default value coming of Colombo from Pub_City. This arise from the constraint using when creating the publisher table by using DEFAULT constraint. The last two rows shows the DEFAULT Constraint of Column Pub_City.

The screenshot shows a SQL query window with the following code:

```
INSERT INTO Publisher ( Pub_ID , Pub_Name , Pub_Country ) 
VALUES ('1013' , 'Sam publishers' , 'India');

INSERT INTO Publisher ( Pub_ID , Pub_Name , Pub_Country , Pub_Contact )
VALUES ('1015' , 'DI publishers' , 'Sri Lanka' , '033145554');

SELECT * FROM Publisher;
```

Below the code, the results of the query are shown in a table:

Pub_ID	Pub_Name	Pub_Country
11	1010	Pantheon house
12	1011	United Kingdom
13	1012	Simon & Shuster
14	1013	Sam publishers
15	1015	India

Then, another table is shown:

Pub_ID	Pub_Name	Pub_Country
12	1011	DC Comics
13	1012	United Kingdom
14	1013	Sam publishers
15	1015	Sri Lanka

Finally, a large table of publisher data is shown:

Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email
3	Macmillans	B003	United States	New York	1999-05-02	0557484871	macmillans123@gmail.com
4	Penguin books	B004	United States		2000-07-20	0118545871	penguinbooks@gmail.com
5	Scholastic press	B005	United Kingdom	Texas	2003-01-26	0715224871	scholasticpress123@gmail.com
6	VI Publishers	B006	Australia	Perth	2005-05-14	0524745871	vipublishers@gmail.com
7	Aspen Publishers	B007	United States		2003-01-26	0715445871	aspenpublishers@hotmail.com
8	Blue Whale Press	B008	United States	New York	2007-02-15	0112857871	bluewhales@gmail.com
9	Scribner press	B009	United States	Chicago	1997-01-16	0047845871	scribnerpress@gmail.com
10	Archid constable and c...	B010	United Kingdom	Bristol	1897-06-26	0769874471	archidconstablecompany@gmail.com
11	Pantheon house	B011	United Kingdom		2000-03-07	0114785371	pantheonhouse123@hotmail.com
12	DC Comics	B012			1900-01-01	022489594	dccomics@yahoo.com
13	Simon & Shuster	B013	United Kingdom	Cardiff	2014-04-15	0112478871	simon&shutter2@gmail.com
14	Sam publishers	NULL	India	Colombo	NULL	NULL	NULL
15	DI publishers	NULL	Sri Lanka	Colombo	NULL	033145554	NULL

Query executed successfully.

LAPTOP

- Updating data using a trigger.

The below is a result of the trigger used to update an already existing value in the table.

```

CREATE TRIGGER Pub_Update
ON Publisher
AFTER UPDATE
AS
    SELECT Pub_ID , Pub_Name , Pub_Country
    FROM Publisher;

    UPDATE Publisher
    SET Pub_Country = 'Sri Lanka'
    WHERE Pub_ID = '1013';
  
```

100 %

	Pub_ID	Pub_Name	Pub_Country
1	1000	Bloomsbury Publishing	United Kingdom
2	1001	United Book Distributors	Australia
3	1002	Macmillans	United States
4	1003	Penguin books	United States
5	1004	Scolastic press	United Kingdom
6	1005	VI Publishers	Australia
7	1006	Aspen Publishers	United States
8	1007	Blue Whale Press	United States
9	1008	Scribner press	United States
10	1009	Archid constable and company	United Kingdom
11	1010	Pantheon house	United Kingdom
12	1011	DC Comics	
13	1012	Simon & Shuster	United Kingdom
14	1013	Sam publishers	Sri Lanka
15	1015	DI publishers	Sri Lanka

- Deletion of an existing trigger

```

DROP TRIGGER IF EXISTS Pub_Insert;
  
```

10 %

Messages

Commands completed successfully.

Completion time: 2021-01-03T18:01:12.1678540+05:30

- Trigger to get the current date

```
/////* trigger to get the date *****/
CREATE TRIGGER date_Update
ON Publisher
AFTER UPDATE
AS
/* TRIGGER TO GET THE CURRENT DATE */
BEGIN
    set nocount on;
    UPDATE publisher set Pub_Date = getdate()
    FROM Publisher p
    INNER JOIN inserted i ON p.Pub_ID = i.Pub_ID
    AND i.Pub_Country = 'Approved'
END
GO

UPDATE Publisher set Pub_Country = 'Approved'
WHERE Pub_ID = 1011
GO

SELECT * FROM Publisher;
```

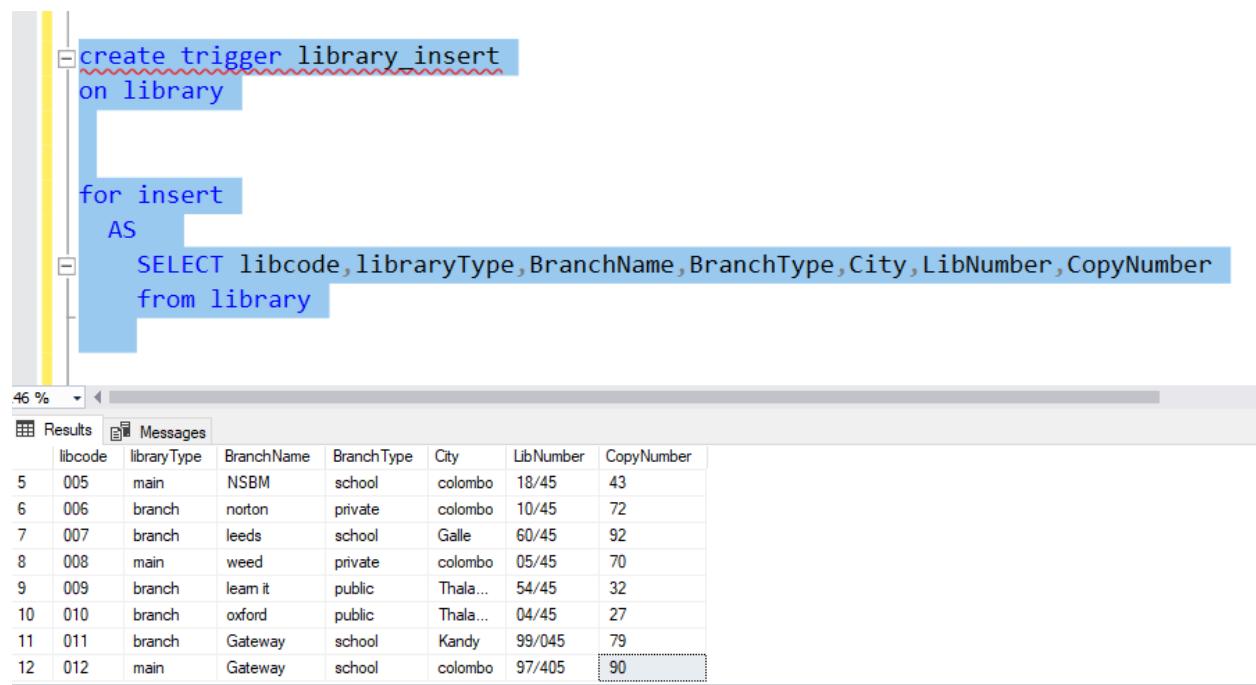
100 %

	Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email
1	1000	Bloomsbury Publishing	B001	United Kingdom	London	1997-06-26	0112445871	bloomsbury.publishing@yahoo.com
2	1001	United Book Distributors	B002	Australia	Sydney	1998-08-15	0225345871	uniteddistributors000@gmail.com
3	1002	Macmillans	B003	United States	New York	1999-05-02	0557484871	macmillans123@gmail.com
4	1003	Penguin books	B004	United States		2000-07-20	0118545871	penguinbooks@ymail.com
5	1004	Scolastic press	B005	United Kingdom	Texas	2003-01-26	0715224871	scolasticpress123@gmail.com
6	1005	VI Publishers	B006	Australia	Perth	2005-05-14	0524745871	vipublishers@gmail.com
7	1006	Aspen Publishers	B007	United States		2003-01-26	0715445871	aspenpublishers@hotmail.com
8	1007	Blue Whale Press	B008	United States	New York	2007-02-15	0112857871	bluewhales@gmail.com
9	1008	Scribner press	B009	United States	Chicago	1997-01-16	0047845871	scibnepress@ymail.com
10	1009	Archid constable and company	B010	United Kingdom	Bristol	1897-06-26	0769874471	archidconstablecompany@gmail.com
11	1010	Pantheon house	B011	United Kingdom		2000-03-07	0114785371	pantheonhouse123@hotmail.com
12	1011	DC Comics	B012	Approved		2021-01-03	022489594	dccomics@yahoo.com
13	1012	Simon & Shuster	B013	United Kingdom	Cardiff	2014-04-15	0112478871	simon&shutter2@gmail.com
14	1013	Sam publishers	NULL	Sri Lanka	Colombo	NULL	NULL	NULL
15	1015	DI publishers	NULL	Sri Lanka	Colombo	NULL	033145554	NULL

- I have used getdate() function to get the date automatically using the trigger. So when entering publisher details in to the table if the published date is not inserted to the table, by using date_update trigger the current date will be added to the published date column automatically.

7) LIBRARY TABLE

Insert trigger



```
create trigger library_insert
on library
for insert
AS
    SELECT libcode,libraryType,BranchName,BranchType,City,LibNumber,CopyNumber
    from library
```

	libcode	libraryType	BranchName	BranchType	City	LibNumber	CopyNumber
5	005	main	NSBM	school	colombo	18/45	43
6	006	branch	norton	private	colombo	10/45	72
7	007	branch	leeds	school	Galle	60/45	92
8	008	main	weed	private	colombo	05/45	70
9	009	branch	team it	public	Thala...	54/45	32
10	010	branch	oxford	public	Thala...	04/45	27
11	011	branch	Gateway	school	Kandy	99/045	79
12	012	main	Gateway	school	colombo	97/405	90

The above sql statements are for the Insert trigger, the new data is inserted to the table by this trigger. As you can see the table initially had only 10 sample data (until libcode 10), now two more (libcode 11&12) are added through the insert trigger.

Update trigger

```
create trigger library_update
on library
for update
as
    SELECT libcode,libraryType,BranchName,BranchType,City,LibNumber,CopyNumber
    from library

create trigger library delete
```

77 %

	libcode	libraryType	BranchName	BranchType	City	LibNumber	CopyNumber
6	006	branch	norton	private	colombo	10/45	72
7	007	branch	leeds	school	Galle	60/45	92
8	008	main	weed	private	colombo	05/45	70
9	009	branch	learn it	public	Thalawathugoda	54/45	32
10	010	branch	oxford	public	Thalawathugoda	04/45	27
11	011	branch	Gateway	school	Kandy	99/045	79
12	012	main	Gateway	school	colombo	97/405	90
13	020	branch	Thanos	public	hikkaduwa	06/07	67

Update trigger changes the table with a new value which is not in the (libcode) correct order as you can see libcode 020 is added after 012.

❖ DATABASE VIEWS

1) BOOK TABLE

1.This is the creation and executing of the view. This view is created to identify the Book by using ISBN. If user types in the ISBN, the book will be seen in the results.

```
*****VIEWS*****
CREATE VIEW View_Book
AS
SELECT *
FROM Book
WHERE ISBN = 'B015';

SELECT * FROM View_Book;
```

	ISBN	B_Name	B_Author	B_PublishedDate	B_Title	B_Category
1	B015	Approved	R. L. Stine	2021-01-03	The Haunted Mask	Horror fiction

2. This is different way of searching category of books.

```
CREATE VIEW View_Book_Novel
AS
SELECT ISBN, B_Name, B_Title, B_Author
FROM Book
WHERE B_Category = 'Novel';

SELECT * FROM View_Book_Novel;
```

	ISBN	B_Name	B_Title	B_Author
1	B001	Harry Potter	and the Philosophers Stone	J. K. Rowling
2	B002	Harry Potter	and the Deathly Hallows	J. K. Rowling
3	B003	Harry Potter	and the Cursed Child	J. K. Rowling
4	B004	Harry Potter	and the Goblet of Fire	J. K. Rowling
5	B005	Harry Potter	and the Order of the Phoenix	J. K. Rowling
6	B006	Harry Potter	and the Chamber of Secrets	J. K. Rowling
7	B007	Harry Potter	and the Prisoner of Azkaban	J. K. Rowling
8	B009	Dracula		Bram Stoker
9	B010	House of leaves		Mark Z. Danielewski
10	B012	The Chronicles	Narnia	C.S.Lewis
11	B013	Twilight	New moon	Stephenie Meyer
12	B019	Twilight	Eclipse	Stephenie Meyer
13	B020	Twilight	Breaking Dawn	Stephenie Meyer

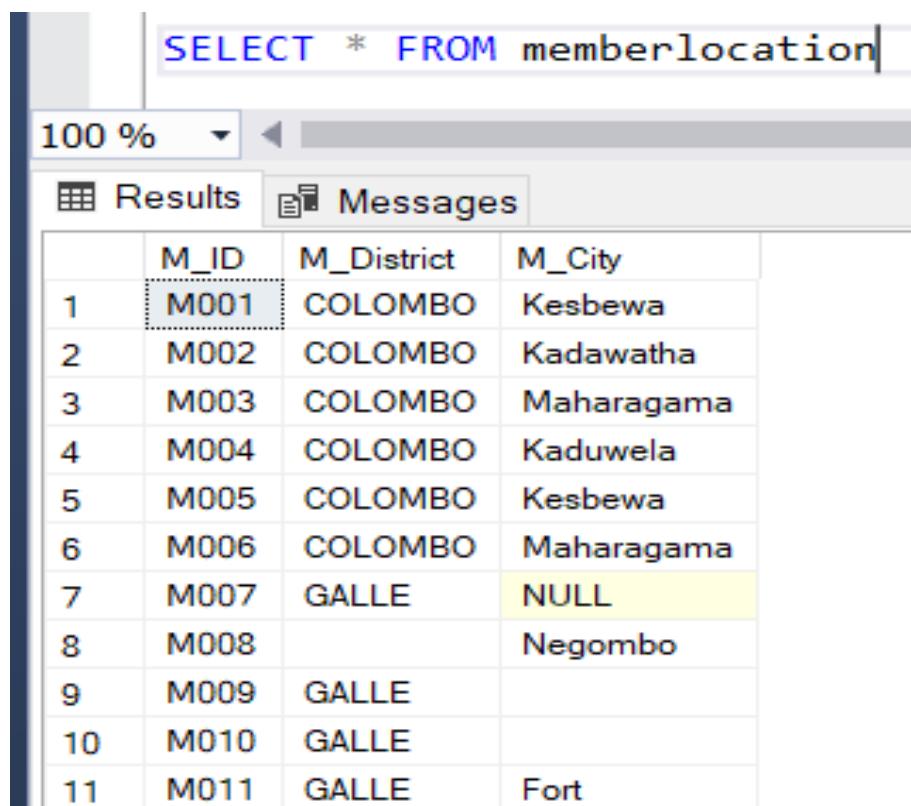
2) BORROWER TABLE

DATABASE VIEWS FOR BORROWER

VIEW TO GET MEMBER LOCATION DETAILS

```
/******VEIWS*****  
/*SQL VIEW TO GET ALL THE MEMBERS LOCATIONS*/  
  
GO  
CREATE VIEW memberlocation AS  
SELECT M_ID, M_District, M_City  
FROM Borrower  
  
SELECT * FROM memberlocation
```

- This view is created in-order to get all members location details which includes his/her Membership ID, District and the City.
- Execution result



The screenshot shows a SQL Server Management Studio window. In the top pane, a query is being typed: `SELECT * FROM memberlocation`. Below the pane, there are two tabs: "Results" and "Messages". The "Results" tab is selected and displays a table with 11 rows of data. The columns are labeled `M_ID`, `M_District`, and `M_City`. The data is as follows:

	M_ID	M_District	M_City
1	M001	COLOMBO	Kesbewa
2	M002	COLOMBO	Kadawatha
3	M003	COLOMBO	Maharagama
4	M004	COLOMBO	Kaduwela
5	M005	COLOMBO	Kesbewa
6	M006	COLOMBO	Maharagama
7	M007	GALLE	NULL
8	M008		Negombo
9	M009	GALLE	
10	M010	GALLE	
11	M011	GALLE	Fort

VIEW TO FIND OUT THE MEMBERS WHO BORROWED BOOKS

```
/*SQL VIEW TO FIND OUT THE MEMBERS WHO HAS BORROWED BOOKS*/
```

```
CREATE VIEW bookborrowed
AS
SELECT TOP 5
M_ID, M_Fname, i.B_ISBN
FROM [dbo].Borrower AS b
INNER JOIN dbo.IssueInfo AS i
ON i.Member_ID = b.M_ID
ORDER BY M_ID ASC

SELECT * FROM bookborrowed
```

- This view is created to find out which borrowers has borrowed books from the issue info table by doing this we get the borrowers member id first name and the ISBN of the book he/she borrowed.
- The Execution result is given below:

The screenshot shows the SQL Server Management Studio interface. In the top query window, the command `SELECT * FROM bookborrowed` is entered. Below it, the results pane is visible, showing the output of the query. The results are presented in a table with three columns: M_ID, M_Fname, and B_ISBN. There are three rows of data, each corresponding to a member who has borrowed books.

	M_ID	M_Fname	B_ISBN
1	M001	Chamoth	B001
2	M002	Neranji	B002
3	M003	Dhanuja	B003

3) COPY TABLE

1. Create View

- Create and execute the ‘CopyViewQuery’ to view books copy information.

The screenshot shows the Object Explorer on the left and the SQL Editor on the right. The SQL Editor contains the following code:

```

/*
*****VIEWS*****
/*CREATE VIEW*/
CREATE VIEW CopyViewbyQuery
AS
SELECT TOP 10 [Copy_ISBN],
CopyNo,
Book_B_Name,
Price AS [Price(Rs.)]
FROM dbo.Copy
INNER JOIN
dbo.[Book] ON
Copy.[Copy_ISBN] = Book.ISBN
ORDER BY Copy.[Copy_ISBN] DESC
SELECT * FROM [dbo].[CopyViewbyQuery]

```

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-VHPDKD".

The results window on the right displays the following data:

	Copy_ISBN	CopyNo	B_Name	Price(Rs.)
1	B020	T01	Twilight	1350
2	B019	S01	Twilight	1630
3	B019	S02	Twilight	1630
4	B018	R01	White House	1490
5	B018	R02	White House	1490
6	B018	R03	White House	1490
7	B018	R04	White House	1490
8	B017	Q01	Goosebumps	1350
9	B017	Q02	Goosebumps	1350
10	B016	P01	Goosebumps	1550

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-VHPDKD".

2. Create View

- Create and execute the ‘PriceView’ to view book price = Rs.1000 related details.

The screenshot shows the Object Explorer on the left and the SQL Editor on the right. The SQL Editor contains the following code:

```

/*
CREATE VIEW - Price*/
CREATE VIEW PriceView
AS
SELECT TOP 15 [Copy_ISBN],
CopyNo,
B.B_Name,
Price AS [Price(Rs.)]
FROM dbo.Copy
INNER JOIN
dbo.[Book] AS B ON
Copy.[Copy_ISBN] = B.ISBN
WHERE Copy.Price = 1000
SELECT * FROM [dbo].[PriceView]

```

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-VHPDKD".

The results window on the right displays the following data:

	Copy_ISBN	CopyNo	B_Name	Price(Rs.)
1	B002	B01	Harry Potter	1000
2	B007	G01	Harry Potter	1000
3	B007	G02	Harry Potter	1000
4	B007	G03	Harry Potter	1000
5	B007	G04	Harry Potter	1000

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-VHPDKD".

4) ISSUE-INFO TABLE

1. Create View

- Create and execute the ‘CreateIssueInfoMemberView’ to view issue information.

The screenshot shows the Object Explorer and the SQL Constraints window. In the SQL Constraints window, the code for the stored procedure is as follows:

```

/*
*****VIEWS*****
*/
/*CREATE VIEW - Issued Book Informations*/
CREATE VIEW CreateIssueInfoMemberView
AS
SELECT TOP 10
    Member_ID,
    B_M_Fname AS [Member Name],
    B_ISBN,
    CopyNo,
    Fine AS [Fine(Rs.)]
FROM dbo.IssueInfo AS Iss
INNER JOIN
    dbo.[Borrower] AS B ON
    B.M_ID = Iss.Member_ID
ORDER BY Member_ID DESC

SELECT * FROM [dbo].[CreateIssueInfoMemberView]

```

The results pane shows the output of the query:

Member_ID	Member Name	B_ISBN	CopyNo	Fine(Rs.)
M020	Last	B007	G02	0
M019	NSBM	B010	J01	0
M017	Hansel	B005	E01	0
M014	Harry	B006	F02	10
M010	Pasindu	B004	D01	0
M008	Manoja	B011	K01	0
M006	Irusha	B008	H01	0
M005	Dasith	B013	M01	0
M004	Samadhi	B009	I01	0
M002	Neranji	B002	B01	10

2. Create View

- Create and execute the ‘CreateFineView’ to view fine <= Rs.10 related details.

The screenshot shows the Object Explorer and the SQL Constraints window. In the SQL Constraints window, the code for the stored procedure is as follows:

```

/*
*****VIEWS*****
*/
/*CREATE VIEW - View Fine Details*/
CREATE VIEW CreateFineView
AS
SELECT TOP 10 B_ISBN,
    Iss.CopyNo,
    Member_ID,
    Borrower_M_Fname AS [Member Name],
    Fine AS [Fine (Rs.)]
FROM dbo.IssueInfo AS Iss
INNER JOIN
    dbo.[Borrower] AS Borrow ON
    Iss.Member_ID = Borrow.M_ID
WHERE Fine <= 10.00

SELECT * FROM [dbo].[CreateFineView]

```

The results pane shows the output of the query:

B_ISBN	CopyNo	Member_ID	Member Name	Fine (Rs.)
B001	A01	M001	Chamoth	10
B002	B01	M002	Neranji	10
B008	H01	M006	Irusha	0
B006	F02	M014	Harry	10
B010	J01	M019	NSBM	0
B004	D01	M010	Pasindu	0
B005	E01	M017	Hansel	0
B007	G02	M020	Last	0
B009	I01	M004	Samadhi	0
B011	K01	M008	Manoja	0

5) STAFF TABLE

1. Create View for ADMIN_INFO without credentials

To find the all information of specified conditions like StaffRole, I used view functions that able to execute so quickly.

```
//SHOW DETAILS OF ADMINS WITHOUT CREDENTIALS  
CREATE VIEW [Admin_INFO] AS  
SELECT FirstName, LastName, StaffID, StaffRole, Contact  
FROM StaffTable  
WHERE StaffRole='Admin';
```

Now this is for find administrators' info without credentials that provided as in AuthSysTable, with this view named Admin_INFO you can seek all info about admins except credentials by using,

```
SELECT * from Admin_INFO;
```

And the result is,

	FirstName	LastName	StaffID	StaffRole	Contact
1	Vindya	Ravindi	11	Admin	732499112

2. Create View for OLD_Members

A view to seek the info about old staff members . In this case old members as staff members who joined the library since more than a year.

```
// SHOW DETAILS OF old members

CREATE VIEW [Old_Members] AS
SELECT FirstName, LastName, StaffID, StaffRole, Contact, Joined_Date
FROM StaffTable
where Joined_Date<'2019-01-01';
```

And the result is showing only old staff members that satisfying above condition.

	FirstName	LastName	StaffID	StaffRole	Contact	Joined_Date
1	Dasitha	Samarasinghe	1	Bookkeeper	772899112	2002-01-19
2	Shevon	Marasinghe	2	Cleaner	772431132	2005-03-15
3	Chamodh	Jayasekara	3	Bookkeeper	772569112	2005-08-23
4	Nethmi	Sulakshika	4	Cleaner	772899432	2006-07-26
5	Dhanuja	Sigera	5	Ledger Holder	776783242	2008-12-12
6	Kevin	Koththigoda	6	Bookkeeper	768392313	2009-01-02
7	Dhanuska	Samarasinghe	7	Bookkeeper	772892312	2009-09-18
8	Pathme	Sirisenage	8	Guard	777829112	2009-09-12
9	Romesh	Udana	9	Cleaner	722499112	2012-03-12
10	Rohitha	Gunaratne	10	Guard	732456722	2014-02-23

6) PUBLISHER TABLE

1. Creating a view

- The below is an example of creation and execution of the view.

The screenshot shows the SQL Server Management Studio interface. In the query window, a CREATE VIEW statement is written:

```
CREATE VIEW View_Publisher
AS
    SELECT * /* CREATION OF A VIEW TO FIND THE PUBLISHER BY THE Pub_ID */
    FROM Publisher
    WHERE Pub_ID = '1005';

SELECT * FROM View_Publisher;
```

Below the query window, the results pane shows a single row of data:

	Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email
1	1005	VI Publishers	B006	Australia	Perth	2005-05-14	0524745871	vipublishers@gmail.com

- This view is created to find the publisher by using the Pub_ID. So if the user type the Pub_ID, publisher will be seen in the results

2. Creation of a view to find the publisher by country

The screenshot shows the SQL Server Management Studio interface. In the query window, a CREATE VIEW statement is written:

```
CREATE VIEW view_PubCountry_UK
AS
    SELECT * /* A VIEW TO FIND THE PUBLISHER FROM THE COUNTRY (e.g. UK) */
    FROM Publisher
    WHERE Pub_Country = 'United kingdom';

SELECT * FROM view_PubCountry_UK;
```

Below the query window, the results pane shows five rows of data:

	Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email
1	1000	Bloomsbury Publishing	B001	United Kingdom	London	1997-06-26	0112445871	bloomsbury.publishing@yahoo.com
2	1004	Scolastic press	B005	United Kingdom	Texas	2003-01-26	0715224871	scolasticpress123@gmail.com
3	1009	Archid constable and company	B010	United Kingdom	Bristol	1897-06-26	0769874471	archidconstablecompany@gmail.com
4	1010	Pantheon house	B011	United Kingdom		2000-03-07	0114785371	pantheonhouse123@hotmail.com
5	1012	Simon & Shuster	B013	United Kingdom	Cardiff	2014-04-15	0112478871	simon&shutter2@gmail.com

- This is another view created to find the publishers by using the country. Here after executing the country all the publishers in the relevant country will be published.

3. Creating a view using multiple tables

The screenshot shows a SQL query window in SSMS. The code creates a view named 'view_toFindBnameAndBauthor' with a single SELECT statement that joins the Book and Publisher tables based on ISBN and Published_BookID respectively. A second SELECT statement is shown below it.

```
CREATE VIEW view_toFindBnameAndBauthor
AS

SELECT Pub_ID , Pub_Name , b.ISBN , B_Name , B_Author , Pub_Date
FROM Book b , Publisher p
WHERE b.ISBN = p.Published_BookID;

SELECT * FROM view_toFindBnameAndBauthor;
```

The results pane displays the data from the view, which consists of 13 rows of book information, including publisher name, ISBN, author name, and publication date.

	Pub_ID	Pub_Name	ISBN	B_Name	B_Author	Pub_Date
1	1000	Bloomsbury Publishing	B001	Harry Potter	J. K. Rowling	1997-06-26
2	1001	United Book Distributors	B002	Harry Potter	J. K. Rowling	1998-08-15
3	1002	Macmillans	B003	Harry Potter	J. K. Rowling	1999-05-02
4	1003	Penguin books	B004	Harry Potter	J. K. Rowling	2000-07-20
5	1004	Scolastic press	B005	Harry Potter	J. K. Rowling	2003-01-26
6	1005	VI Publishers	B006	Harry Potter	J. K. Rowling	2005-05-14
7	1006	Aspen Publishers	B007	Harry Potter	J. K. Rowling	2003-01-26
8	1007	Blue Whale Press	B008	Cinderella	Daisy Fisher	2007-02-15
9	1008	Scribner press	B009	Dracula	Bram Stoker	1997-01-16
10	1009	Archid constable and company	B010	House of leaves	Mark Z. Danielewski	1897-06-26
11	1010	Pantheon house	B011	Wonder Woman	Leigh Bardugo	2000-03-07
12	1011	DC Comics	B012	The Chronicles	C.S.Lewis	2021-01-03
13	1012	Simon & Shuster	B013	Twilight	Stephenie Meyer	2014-04-15

- This view is created by using the publisher and the book tables.
- I have used joins here to create them.
- This basically shows the Pub_ID, Pub_Name, ISBN, B_Name, B_Author, and the Pub_Date.
- This can be used to find which author published which book rather than comparing Published_BookID in the publisher table and ISBN in the boom table.

7) LIBRARY TABLE

```
CREATE view  
BranchAddress AS  
SELECT City, LibNumber,  
FROM library  
WHERE City = 'colombo';  
  
select* from BranchAddress
```

77 %

Results Messages

	City	LibNumber
2	colombo	11/45
3	colombo	18/45
4	colombo	10/45
5	colombo	05/45

The above sql statements are used to create a view which filters to get the BranchAddress of the libraries in the city colombo from the entered values BranchAddress is the name of the view and this shows only the values from (city) &(LibNumber) since those are the only two selected values.

```
CREATE VIEW  
lib_first3  
as  
SELECT libcode,BranchName, BranchType , LibraryType  
FROM library  
where libcode<=3  
select * FROM lib_first3
```

177 %

Results Messages

libcode	BranchName	BranchType	LibraryType
1 001	nex	public	main
2 002	IBT	school	Branch
3 003	NSBM	school	Branch

❖ USER DEFINED FUNCTIONS

1) BOOK TABLE

➤ USER DEFINED FUN

```
/*************FUNCTIONS*****/
CREATE FUNCTION B_Category(@BCategory varchar (50))
RETURNS TABLE
AS
RETURN
(SELECT ISBN, B_Name, B_PublishedDate, B_Title
FROM Book
WHERE B_Category = @BCategory);

SELECT * FROM B_Category('Novel');
```

The screenshot shows the results of the query. A table is displayed with columns: ISBN, B_Name, B_PublishedDate, and B_Title. The data includes books from the Harry Potter series and other novels.

	ISBN	B_Name	B_PublishedDate	B_Title
1	B001	Harry Potter	2021-01-03	and the Philosophers Stone
2	B002	Harry Potter	2021-01-03	and the Deathly Hallows
3	B003	Harry Potter	2021-01-03	and the Cursed Child
4	B004	Harry Potter	2021-01-03	and the Goblet of Fire
5	B005	Harry Potter	2021-01-03	and the Order of the Phoenix
6	B006	Harry Potter	2021-01-03	and the Chamber of Secrets
7	B007	Harry Potter	2021-01-03	and the Prisoner of Azkaban
8	B009	Dracula	2021-01-03	
9	B010	House of leaves	2021-01-03	
10	B012	The Chronicles	2021-01-03	Narnia
11	B013	Twilight	2021-01-03	New moon

Query executed successfully.

- This function is to identify the book whose category is Novel.
- This is a scalar function.

```
CREATE FUNCTION B_Details (@id varchar(5))
RETURNS TABLE
AS
RETURN
(SELECT ISBN, B_Name, B_Author
FROM Book
WHERE ISBN = @id)

SELECT * FROM B_Details ('B016');
```

The screenshot shows the results of the query. A table is displayed with columns: ISBN, B_Name, and B_Author. The data shows the details for the book with ISBN B016.

	ISBN	B_Name	B_Author
1	B016	Goosebumps	R. L. Stine

This function is to find the name and author of the book after entering the ISBN.

This is the result as shown above.

2) BORROWER TABLE

USER DEFINED FUNCTIONS

FUNCTION TO FIND OUT THE COUNT OF MEMBERS IN A DISRTICT

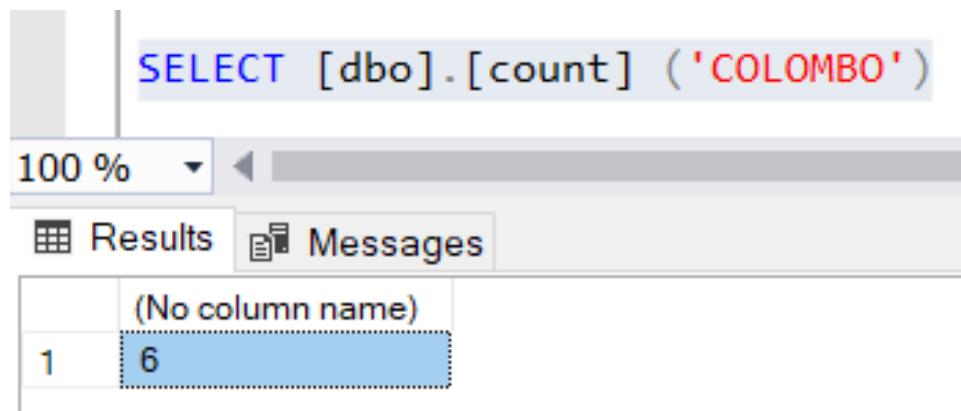
```
*****FUNCTIONS*****
```

```
/*SQL FUNTION TO FIND HOW MANY MEMBERS ARE IN A PARTICULAR DISTRICT*/
```

```
CREATE FUNCTION [dbo].[count] (@mdistrict varchar (20))
RETURNS int
AS
BEGIN
DECLARE @count int
SELECT @count = COUNT (M_ID)
FROM Borrower
WHERE M_District = @mdistrict
RETURN @count
END
```

```
SELECT [dbo].[count] ('COLOMBO')
```

- By using this function, we can find out the count of members who comes from a particular district a value needs to be passed either its COLOMBO, GALLE OR JAFFNA. Then the count will be displayed accordingly.
- Execution are as follows



The screenshot shows a SQL query window in SSMS. The query is:

```
SELECT [dbo].[count] ('COLOMBO')
```

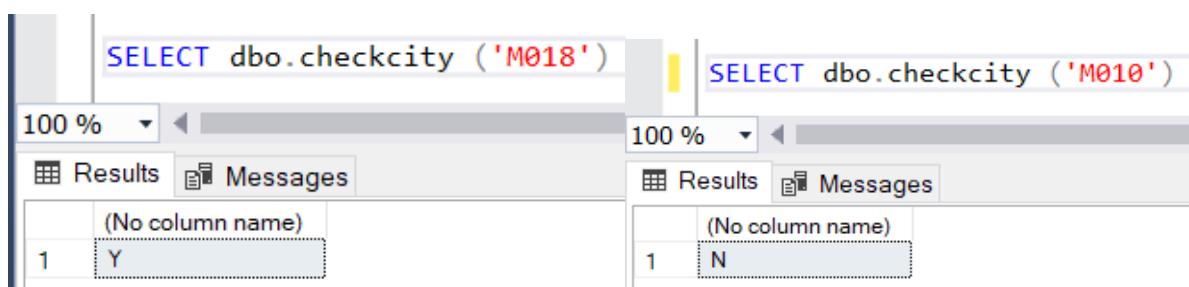
The results pane shows a single row with the value 6.

(No column name)
1 6

FUNCTION TO CHECK WHETHER A PERSON HAS A CITY OR NOT

```
/*FUNCTION TO CHECK WHETHER A GIVENS MEMBERS CITY IS NULL OR NOT*/  
  
CREATE FUNCTION checkcity (@mid varchar(4))  
RETURNS varchar  
AS  
BEGIN  
DECLARE @city varchar (20)  
DECLARE @result varchar(3)  
SELECT @city = M_City  
FROM Borrower  
WHERE M_ID = @mid  
  
IF @city IS NULL  
BEGIN  
SET @result = 'Yes'  
end  
Else  
begin  
set @result = 'No'  
END  
RETURN @result  
END  
  
SELECT dbo.checkcity ('M018')
```

- By using this function we can find out the whether a given members city is given or NULL this can be simply done by passing the members Membership ID into the statement and if its Null its YES and if its NOT NULL then NO.
- Execution are as follows.



The screenshot shows two separate SSMS windows. Both windows have a title bar with '100 %' and a dropdown arrow. The left window has a query editor with the text 'SELECT dbo.checkcity ('M018')'. Below it is a results grid with one row: column 1 contains '1' and column 2 contains 'Y'. The right window also has a query editor with the same query. Below it is a results grid with one row: column 1 contains '1' and column 2 contains 'N'.

3) COPY TABLE

1. Create SQL Scalar Functions.

- Create and execute the 'SummationOfBookPrice' function, to take Summation of book prices.

The screenshot shows the Object Explorer and the SQL Editor in SSMS. The Object Explorer displays various database objects like Tables, Views, and Functions. The SQL Editor contains the following code:

```

/*
*****FUNCTIONS*****
*/
CREATE FUNCTION SummationOfBookPrice (@Copy_ISBN int)
RETURNS FLOAT
AS
BEGIN
    RETURN (SELECT SUM([Price]) FROM [Copy]
    WHERE Copy.[Copy_ISBN] = @Copy_ISBN)
END

SELECT [Copy_ISBN],
    SUM([Price]) as [Total Price of the Books(Rs.)]
FROM [Copy]
group by [Copy_ISBN]

```

The Messages pane shows "Commands completed successfully." and the Completion time: 2021-01-02T21:05:08.6883815+05:30. The Results pane displays the following data:

Copy_ISBN	Total Price of the Books(Rs.)
B001	1115
B002	1000
B003	4500
B004	1750
B005	5900
B006	1350
B007	4000
B008	3000
B009	1350
B010	1750
B011	1975
B012	1050
B013	1100
B014	4500
B015	1750
B016	4650
B017	2700
B018	5960
B019	3260
B020	1350

2. Create SQL Scalar Functions.

- Create and execute the 'LibraryCopy' function.

The screenshot shows the Object Explorer and the SQL Editor in SSMS. The Object Explorer displays various database objects. The SQL Editor contains the following code:

```

/*
SQL Inline Function with Parameters
*/
CREATE FUNCTION LibraryCopy (@copyNo VARCHAR(10))
RETURNS TABLE
AS
RETURN (
    SELECT [CopyNo],
        [Copy_ISBN],
        B.B_Name AS BookName,
        [Price] AS [Price(Rs.)]
    FROM [Copy]
    INNER JOIN
    Book AS B ON
    B.[ISBN] = [Copy].Copy_ISBN
    WHERE [CopyNo] = @copyNo
)

```

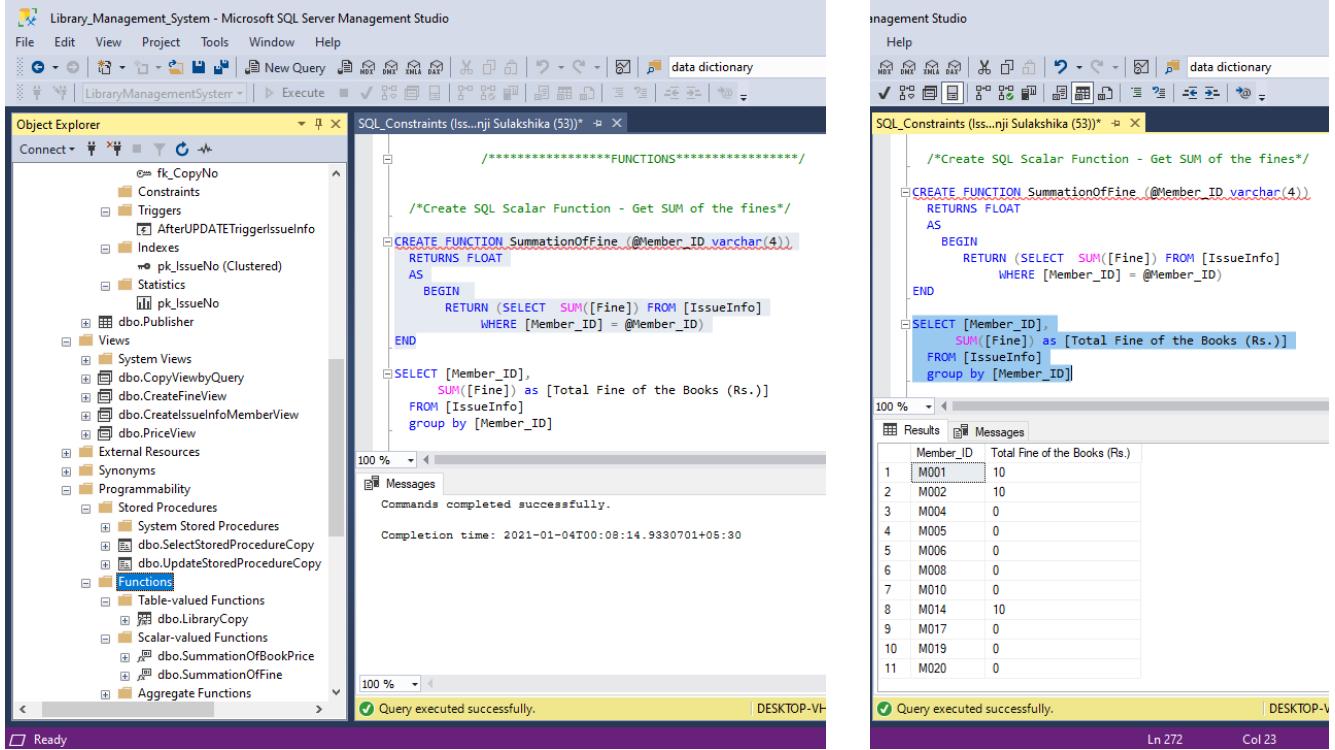
The Messages pane shows "Commands completed successfully." and the Completion time: 2021-01-01T16:38:50.5619489+05:30. The Results pane displays the following data:

CopyNo	Copy_ISBN	BookName	Price(Rs.)
A01	B001	Harry Potter	1115

4) ISSUE-INFO TABLE

1. Create SQL Scalar Functions.

- Create and execute the 'SummationOffine' function, to take summation of the fine.

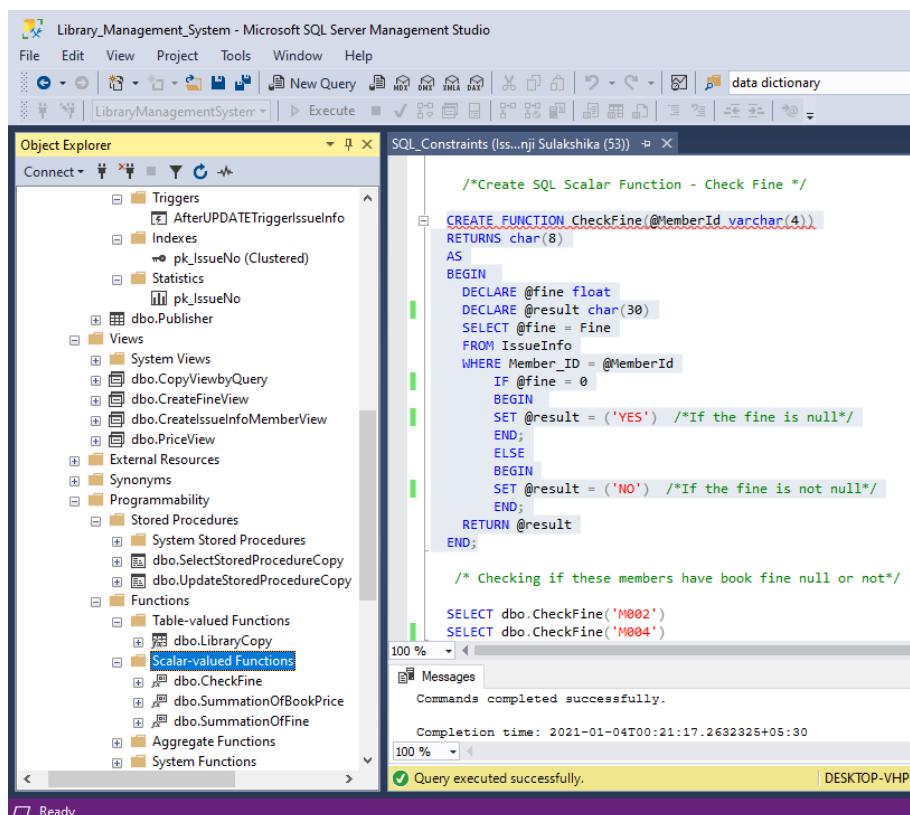


The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure, including tables like 'IssueInfo' and 'MemberInfo', and various stored procedures and functions. The central pane shows the Transact-SQL code for creating a scalar function named 'SummationOffine'. The code uses a SELECT statement with a GROUP BY clause to calculate the total fine for a given member ID. Below the code, the 'Messages' pane shows a success message: 'Commands completed successfully.' and the completion time: 'Completion time: 2021-01-04T00:08:14.9330701+05:30'. The bottom status bar indicates 'Query executed successfully.' and the session details: 'Ln 272' and 'Col 23'. To the right, another window shows the results of the query, displaying a table with 'Member_ID' and 'Total Fine of the Books (Rs.)' columns. The data is as follows:

Member_ID	Total Fine of the Books (Rs.)
M001	10
M002	10
M004	0
M005	0
M006	0
M008	0
M010	0
M014	10
M017	0
M019	0
M020	0

2. Create SQL Scalar Functions.

- Create and execute the 'CheckFine' function, to display fine = 0 or not.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows the database structure. The central pane contains the Transact-SQL code for creating a scalar function named 'CheckFine'. The function takes a member ID as input and returns a character value ('YES' or 'NO') indicating if the fine is null or not. Below the code, the 'Messages' pane shows a success message: 'Commands completed successfully.' and the completion time: 'Completion time: 2021-01-04T00:21:17.2632325+05:30'. The bottom status bar indicates 'Query executed successfully.' and the session details: 'Ln 272' and 'Col 23'. The right side of the screen shows the results of the query, which are not visible in the screenshot but are mentioned in the text below.

Management Studio

```

/*Create SQL Scalar Function - Check Fine */

CREATE FUNCTION CheckFine(@MemberId varchar(4))
RETURNS char(8)
AS
BEGIN
    DECLARE @fine float
    DECLARE @result char(30)
    SELECT @fine = Fine
    FROM IssueInfo
    WHERE Member_ID = @MemberId
    IF @fine = 0
    BEGIN
        SET @result = ('YES') /*If the fine is null*/
    END;
    ELSE
    BEGIN
        SET @result = ('NO') /*If the fine is not null*/
    END;
    RETURN @result
END;

/* Checking if these members have book fine null or not*/

SELECT dbo.CheckFine('M002')
SELECT dbo.CheckFine('M004')

```

100 %

Results Messages

(No column name)
1 NO

Query executed successfully.

DESKTOP-VHPI

Ln 301 Col 1 Ch

Management Studio

```

/*Create SQL Scalar Function - Check Fine */

CREATE FUNCTION CheckFine(@MemberId varchar(4))
RETURNS char(8)
AS
BEGIN
    DECLARE @fine float
    DECLARE @result char(30)
    SELECT @fine = Fine
    FROM IssueInfo
    WHERE Member_ID = @MemberId
    IF @fine = 0
    BEGIN
        SET @result = ('YES') /*If the fine is null*/
    END;
    ELSE
    BEGIN
        SET @result = ('NO') /*If the fine is not null*/
    END;
    RETURN @result
END;

/* Checking if these members have book fine null or not*/

SELECT dbo.CheckFine('M002')
SELECT dbo.CheckFine('M004')

```

100 %

Results Messages

(No column name)
1 YES

Query executed successfully.

DESKTOP-VHPI

Ln 302 Col 1 Ch

5) STAFF TABLE

1. Create a function for search information just by Staff ID

Creating functions can make yourself easy to search information just hitting by a desired key like an example as Staff ID,

```
CREATE FUNCTION findname (@id int)
RETURNS VARCHAR (50)
AS
BEGIN
DECLARE @name VARCHAR(50)
SELECT @name = CONCAT(FirstName, ' ', LastName)
FROM StaffTable
WHERE StaffID = @id
RETURN @name
END
```

After creating this, a member can enter just by Staff ID to load any of details of specified person by using following statement.

```
SELECT dbo.findname(5) AS [Staff Member Name]
```

So, the Staff ID is 5 as, we are about to find the full name by that desired staff id, and therefore the result like this,

	Staff Member Name
1	Dhanuja Sigera

2. Create a function to search the joined date just by Staff ID

Finally, I created another function to perform the execution of joined date of a staff members just entered within staff id,

```
CREATE FUNCTION findjoineddate (@id int)
returns date
AS
BEGIN
DECLARE @date date
SELECT @date = Joined_Date FROM StaffTable
where @id = StaffID
return @date
END
```

And execution part can be done through this,

```
SELECT dbo.findjoineddate(7) AS [Staff Member Joined Date]
```

As to find the joined date of staff member who represent by staff id ‘7’ can be displayed like this,

Results	
Messages	
1	2009-09-18

6) PUBLISHER TABLE

- Creating a function

The screenshot shows a SQL query window with the following code:

```
CREATE FUNCTION Pub_Details (@id varchar(5))
RETURNS TABLE
AS
RETURN
(SELECT Pub_ID , Pub_Name, Pub_Contact
FROM Publisher
WHERE Pub_ID = @id)

SELECT * FROM Pub_Details ('1005');
```

The results pane shows a single row of data:

	Pub_ID	Pub_Name	Pub_Contact
1	1005	VI Publishers	0524745871

- This is the result after creating and executing the function.
- This function is used to find the publisher name and the published country after entering the publisher id.
- This is an easy way to access the publisher details by using the publisher id.

- Creating a function using the Pub_Country

The screenshot shows a SQL query window with the following code:

```
CREATE FUNCTION Pub_Country (@PCountry varchar (50))
RETURNS TABLE
AS
RETURN
(SELECT Pub_ID , Pub_Name , Published_BookID , Pub_Contact , Pub_Country , Pub_Date
FROM Publisher
WHERE Pub_Country = @PCountry);

SELECT * FROM Pub_Country('United States');
```

The results pane shows four rows of data:

	Pub_ID	Pub_Name	Published_BookID	Pub_Contact	Pub_Country	Pub_Date
1	1002	Macmillans	B003	0557484871	United States	1999-05-02
2	1003	Penguin books	B004	0118545871	United States	2000-07-20
3	1007	Blue Whale Press	B008	0112857871	United States	2007-02-15
4	1008	Scribner press	B009	0047845871	United States	1997-01-16

- The above is a scalar function.
- I have created this function to find the publishers who lives in United States.
- By using this function we can find the publishers who lives in US and their information.

❖ STORED PROCEDURES

1) BOOK TABLE

➤ STORED PROCEDURES

The screenshot shows the SQL Server Management Studio interface. In the main pane, a script is displayed:

```
/******PROCEDURES*****/
CREATE PROCEDURE find_book
@id varchar (5)
AS
SELECT ISBN, B_Name, B_Title
FROM Book
WHERE ISBN = @id      /****Create****/

EXEC find_book 'B016';    /****Execute***/
EXEC find_book 'B017';
```

The results pane shows two rows of data from the Book table:

	ISBN	B_Name	B_Title
1	B016	Goosebumps	Welcome to Dead House
1	B017	Goosebumps	Night of the Living Dummy

By searching the ISBN, we can get the information of name and title of the book.

The screenshot shows the SQL Server Management Studio interface. In the main pane, a script is displayed:

```
CREATE PROCEDURE noOfBOOK
@ISBN varchar(5)
AS
BEGIN
DECLARE @totbook int
SELECT @totbook = count(ISBN)FROM
Book
RETURN @totbook
END

DECLARE @b_value int
EXEC @b_value = noOfBOOK ''
PRINT @b_value
```

The results pane shows the output of the stored procedure:

	Messages
1	20

Completion time: 2021-01-03T21:26:14.7979850+05:30

This is to find out how many book records are there in total.

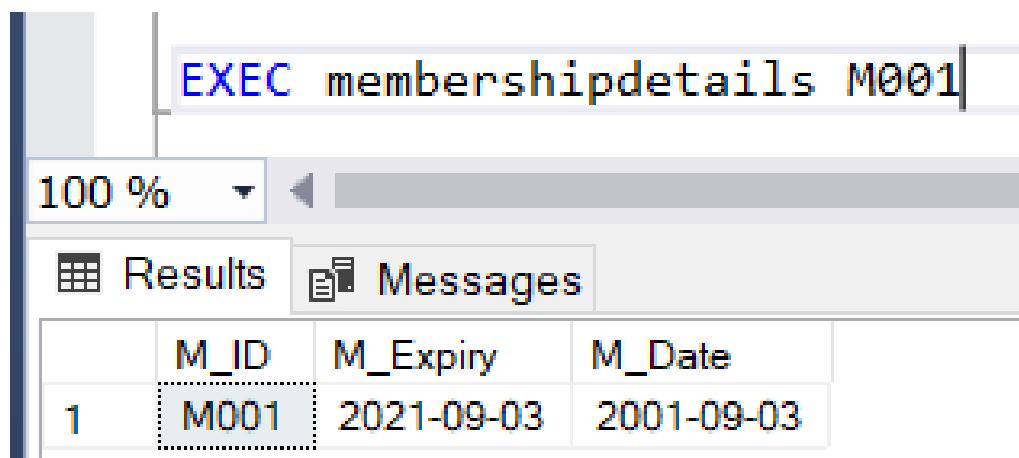
2) BORROWER TABLE

STORED PROCEDURES

PROCEDURE TO GET MEMBERSHIP INFORMATION DETAILS

```
/*SQL PROCEDURE TO GET MEMBERSHIP INFORMATION OF A GIVEN MEMBER*/  
  
CREATE Procedure membershipdetails  
(@mid varchar(4))  
AS  
BEGIN  
SELECT M_ID, M_Expiry, M_Date  
FROM Borrower  
WHERE M_ID = @mid  
END  
  
EXEC membershipdetails M001
```

- We can use this procedure to input a members ID and get his/her membership details as it showcase the members joined date and membership expiry date.
- Execution details are as follows:



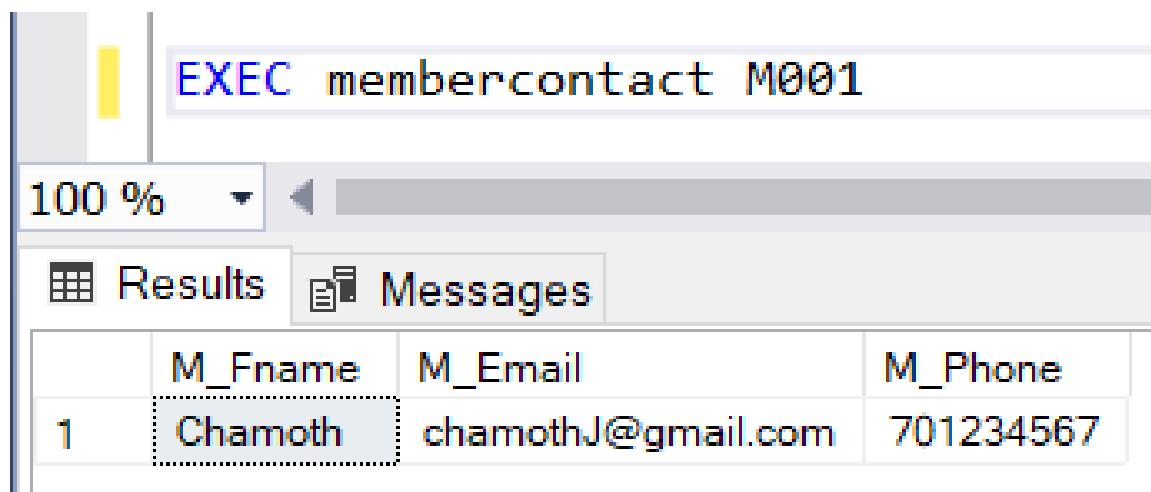
The screenshot shows the SQL Server Management Studio interface. In the query window, the command `EXEC membershipdetails M001` is typed. Below the query window, the results tab is selected, displaying a table with three columns: M_ID, M_Expiry, and M_Date. The data row shows values: 1, M001, 2021-09-03, and 2001-09-03 respectively.

	M_ID	M_Expiry	M_Date
1	M001	2021-09-03	2001-09-03

PROCEDURE TO GET CONTACT INFORMATION OF A MEMBER

```
/*****PROCEDURES*****  
/*SQL PROCEDURE TO GET CONTACT INFORMATION OF A GIVEN MEMBER*/  
  
CREATE Procedure membercontact  
(@mid varchar(4))  
AS  
BEGIN  
SELECT M_Fname, M_Email, M_Phone  
FROM Borrower  
WHERE M_ID = @mid  
END  
  
EXEC membercontact M006
```

- We can use this procedure to find out a particular member contact information so when a membership ID is being passed the related member will be displayed along with the email address and telephone number of the borrower.



The screenshot shows the SQL Server Management Studio interface. In the query window, the command `EXEC membercontact M001` is entered. Below the query window, the results pane is visible, showing a table with three columns: `M_Fname`, `M_Email`, and `M_Phone`. There is one row returned, with the values `Chamoth`, `chamothJ@gmail.com`, and `701234567`.

	M_Fname	M_Email	M_Phone
1	Chamoth	chamothJ@gmail.com	701234567

3) COPY TABLE

1. Select Stored Procedure.

- Create and execute the ‘SelectStoredProcedure’.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The left pane, titled 'Object Explorer', displays the database schema for 'LibraryManagementSystem'. It includes tables like 'dbo.Book' and 'dbo.Borrower', and a table 'dbo.Copy' with columns 'Copy_ISBN', 'CopyNo', and 'Price'. The right pane, titled 'SQL_Constraints (Co...ji Sulakshika (57))*', contains a T-SQL script for creating a stored procedure. The script starts with a comment block for procedures and then defines a stored procedure named 'SelectStoredProcedureCopy'. It uses 'SET NOCOUNT ON' and two 'SELECT TOP 15' statements to retrieve data from the 'Copy' table, once in ascending order by price and once in descending order. The 'Messages' pane at the bottom indicates that the command was completed successfully. The status bar at the bottom right shows the computer name 'DESKTOP-VHPDJKD' and the completion time '2021-01-01T16:28:47.2033502+05:30'.

```
SQL_Constraints (Copy).sql - DESKTOP-VHPDJKD.LibraryManagementSystem (DESKTOP-VHPDJKD\Neranji Sulakshika (57)) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help
New Query Execute
Object Explorer
SQL_Constraints (Co...ji Sulakshika (57))
PROCEDURES
/*
**SELECT STORED PROCEDURE**
IF OBJECT_ID ('SelectStoredProcedureCopy') IS NOT NULL
DROP PROCEDURE SelectStoredProcedureCopy;

CREATE PROCEDURE SelectStoredProcedureCopy
AS
BEGIN
    SET NOCOUNT ON;
    SELECT TOP 15 [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
    FROM [Copy]
    ORDER BY [Price] ASC

    SELECT TOP 15 [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
    FROM [Copy]
    ORDER BY [Price] DESC
END
*/
100 %
Messages
Commands completed successfully.
Completion time: 2021-01-01T16:28:47.2033502+05:30
100 %
Query executed successfully. DESKTOP-VHPDJKD (15.
Ready
```

- Creating and executing the select stored procedure.

I. Execute the select stored procedure order by Price with ASC order.

```

CREATE PROCEDURE SelectStoredProcedureCopy
AS
BEGIN
    SET NOCOUNT ON;
    SELECT TOP 15 [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
      FROM [Copy]
     ORDER BY [Price] ASC
END
  
```

	Copy_ISBN	CopyNo	Price(Rs.)
1	B007	G04	1000
2	B007	G03	1000
3	B007	G02	1000
4	B007	G01	1000
5	B002	B01	1000
6	B001	A01	1050
7	B012	L01	1050
8	B013	M01	1100
9	B006	F02	1350
10	B009	I01	1350
11	B017	Q01	1350
12	B017	Q02	1350
13	B020	T01	1350
14	B018	R01	1490
15	B018	R02	1490

Query executed successfully.

Execute the select stored procedure procedure order by CopyNo with ASC & DESC order.

II. Execute the select stored procedure order by Price with DESC order.

```

SELECT TOP 15 [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
      FROM [Copy]
     ORDER BY [Price] DESC
  
```

	Copy_ISBN	CopyNo	Price(Rs.)
1	B005	E03	1975
2	B005	E02	1975
3	B011	K01	1975
4	B005	E01	1950
5	B004	D01	1750
6	B010	J01	1750
7	B015	O01	1750
8	B019	S01	1630
9	B019	S02	1630
10	B016	P01	1550
11	B016	P02	1550
12	B016	P03	1550
13	B003	C03	1500
14	B003	C02	1500
15	B003	C01	1500

Query executed successfully.

IV. Execute the select stored procedure order by Copy_ISBN with ASC order.

```

SELECT TOP 10 [Copy_ISBN],
           [CopyNo]
      FROM [Copy]
     ORDER BY [CopyNo] ASC

SELECT TOP 5 [Copy_ISBN],
           [CopyNo]
      FROM [Copy]
     ORDER BY [CopyNo] DESC
  
```

	Copy_ISBN	CopyNo
1	B001	A01
2	B002	B01
3	B003	C01
4	B003	C02
5	B003	C03
6	B004	D01
7	B005	E01
8	B005	E02
9	B005	E03
10	B006	F02

	Copy_ISBN	CopyNo
1	B020	T01
2	B019	S02
3	B019	S01
4	B018	R04
5	B018	R03

Query executed successfully.

```

SELECT TOP 15 [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
      FROM [Copy]
     ORDER BY [Copy_ISBN] ASC
  
```

	Copy_ISBN	CopyNo	Price(Rs.)
1	B001	A01	1050
2	B002	B01	1000
3	B003	C03	1500
4	B003	C02	1500
5	B003	C01	1500
6	B004	D01	1750
7	B005	E03	1975
8	B005	E02	1975
9	B005	E01	1950
10	B006	F02	1350
11	B007	G04	1000
12	B007	G03	1000
13	B007	G02	1000
14	B007	G01	1000
15	B008	H02	1500

Query executed successfully.

- V. Execute the select stored procedure procedure order by Price with ASC order.

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window. The title bar reads "ManagementSystem (DESKTOP-VHPDJKD\Neranji Sulakshika (57)) - Microsoft SQL Server Management Studio". The query editor window contains the following T-SQL code:

```
SELECT TOP 10 [Copy_ISBN],  
       [CopyNo],  
       [Price] AS [Price(Rs.)]  
  FROM [Copy]  
 ORDER BY [Copy_ISBN] DESC  
END  
  
EXEC [dbo].[SelectStoredProcedureCopy]
```

The results pane displays three tables of data:

	Copy_ISBN	CopyNo
1	B001	A01
4	B018	R04

	Copy_ISBN	CopyNo	Price(Rs.)
1	B001	A01	1050

	Copy_ISBN	CopyNo	Price(Rs.)
3	B019	S02	1630
4	B018	R01	1490
5	B018	R02	1490
6	B018	R03	1490
7	B018	R04	1490
8	B017	Q01	1350
9	B017	Q02	1350
10	B016	P01	1550

At the bottom of the results pane, a message says "Query executed successfully." with a green checkmark icon. The status bar at the bottom right shows "Ln 157" and "Col 1".

2. Update Stored Procedure.

- Creating and executing the update stored procedure.

The screenshot shows two side-by-side windows in Microsoft SQL Server Management Studio. The left window displays the Object Explorer with the 'Programmability' node expanded, showing 'Stored Procedures'. The right window shows the 'SQL Constraints' query editor with the following code:

```

/*UPDATE STORED PROCEDURE*/
IF OBJECT_ID ('UpdateStoredProcedureCopy') IS NOT NULL
    DROP PROCEDURE UpdateStoredProcedureCopy;

CREATE PROCEDURE UpdateStoredProcedureCopy
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE [Copy] SET [Price] = 1115.00
    WHERE [CopyNo] = 'A01';

    SELECT * FROM Copy WHERE CopyNo= 'A01';

    EXEC [dbo].[UpdateStoredProcedureCopy]

    SELECT [Copy_ISBN],
           [CopyNo],
           [Price] AS [Price(Rs.)]
      FROM [Copy]
     ORDER BY [Price] DESC

```

The status bar at the bottom of the right window indicates "Query executed successfully." and "Ln 176 Col 1 Ch 1".

- I. Execute the select stored procedure procedure order by Price with DESC order.

The screenshot shows the 'SQL Constraints' query editor with the following code:

```

SELECT [Copy_ISBN],
       [CopyNo],
       [Price] AS [Price(Rs.)]
  FROM [Copy]
 ORDER BY [Price] DESC

```

The results pane shows the following table:

	Copy_ISBN	CopyNo	Price(Rs.)
1	B005	E02	1975
2	B005	E03	1975
3	B011	K01	1975
4	B005	E01	1950
5	B004	D01	1750
6	B010	J01	1750
7	B015	O01	1750
8	B019	S01	1630
9	B019	S02	1630
10	B016	P01	1550
11	B016	P02	1550
12	B016	P03	1550
13	B014	N01	1500
14	B014	N02	1500
15	B014	N03	1500
16	B008	H01	1500
17	B008	H02	1500
18	B003	C01	1500
19	B003	C02	1500
20	B003	C03	1500

The status bar at the bottom of the right window indicates "Query executed successfully." and "Ln 181".

4) ISSUE-INFO TABLE

1. Select Stored Procedure.

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure, including keys, triggers, indexes, and stored procedures. The central pane shows the T-SQL code for creating and executing a stored procedure. The right pane shows the results of the executed query.

```

CREATE PROCEDURE SelectStoredProcedureIssueInfo
AS
BEGIN
    SET NOCOUNT ON;
    SELECT TOP 10 [IssueNo],
           [B_ISBN],
           [CopyNo],
           [Member_ID],
           [Fine] AS [Fine (Rs.)],
           [IssueDate],
           [ReturnDate]
      FROM [IssueInfo]
     ORDER BY [B_ISBN] ASC
END

```

Execution Results:

```

SELECT TOP 10 [IssueNo],
           [B_ISBN],
           [CopyNo],
           [Member_ID],
           [Fine] AS [Fine (Rs.)],
           [IssueDate],
           [ReturnDate]
      FROM [IssueInfo]
     ORDER BY [CopyNo] DESC

```

```

SELECT TOP 10 [IssueNo],
           [B_ISBN],
           [CopyNo],
           [Member_ID],
           [Fine] AS [Fine (Rs.)],
           [IssueDate],
           [ReturnDate]
      FROM [IssueInfo]
     ORDER BY [Member_ID] ASC

```

```

SELECT TOP 10 [IssueNo],
           [B_ISBN],
           [CopyNo],
           [Member_ID],
           [Fine] AS [Fine (Rs.)],
           [IssueDate],
           [ReturnDate]
      FROM [IssueInfo]
     ORDER BY [IssueDate] DESC

```

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	B001	A01	M001	10	2020-12-18	2021-01-02
2	B002	B01	M002	10	2020-12-18	2021-01-01
3	B004	D01	M010	0	2020-12-25	2021-01-08
4	B005	E01	M017	0	2020-12-27	2021-01-10
5	B006	F02	M014	10	2020-12-23	2021-01-06
6	B007	G02	M020	0	2020-12-27	2021-01-10
7	B008	H01	M006	0	2020-12-23	2021-01-06
8	B009	I01	M004	0	2020-12-31	2021-01-14

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
12	B013	M01	M005	0	2021-01-01	2021-01-15
11	B011	K01	M008	0	2021-01-01	2021-01-15
6	B010	J01	M019	0	2020-12-24	2021-01-07
10	B009	I01	M004	0	2020-12-31	2021-01-14

- Creating and executing the select stored procedure.

I. Execute the select stored procedure order by B_ISBN with ASC order.

```

CREATE PROCEDURE SelectStoredProcIssueInfo
AS
BEGIN
    SET NOCOUNT ON;
    SELECT TOP 10 [IssueNo],
           [B_ISBN],
           [CopyNo],
           [Member_ID],
           [Fine] AS [Fine (Rs.)],
           [IssueDate],
           [ReturnDate]
      FROM [IssueInfo]
     ORDER BY [B_ISBN] ASC
END

```

The screenshot shows the results of the query in a table:

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	B001	A01	M001	10	2020-12-18	2021-01-02
2	B002	B01	M002	10	2020-12-18	2021-01-01
3	B004	D01	M010	0	2020-12-25	2021-01-08
4	B005	E01	M017	0	2020-12-27	2021-01-10
5	B006	F02	M014	10	2020-12-23	2021-01-06
6	B007	G02	M020	0	2020-12-27	2021-01-10
7	B008	H01	M006	0	2020-12-23	2021-01-06
8	B009	I01	M004	0	2020-12-31	2021-01-14
9	B010	J01	M019	0	2020-12-24	2021-01-07
10	B011	K01	M008	0	2021-01-01	2021-01-15

Query executed successfully.

II. Execute the select stored procedure order by CopyNo with DESC order.

```

SELECT TOP 10 [IssueNo],
       [B_ISBN],
       [CopyNo],
       [Member_ID],
       [Fine] AS [Fine (Rs.)],
       [IssueDate],
       [ReturnDate]
  FROM [IssueInfo]
 ORDER BY [CopyNo] DESC

```

The screenshot shows the results of the query in a table:

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	B013	M01	M005	0	2021-01-01	2021-01-15
2	B011	K01	M008	0	2021-01-01	2021-01-15
3	B010	J01	M019	0	2020-12-24	2021-01-07
4	B009	I01	M004	0	2020-12-31	2021-01-14
5	B008	H01	M006	0	2020-12-23	2021-01-06
6	B007	G02	M020	0	2020-12-27	2021-01-10
7	B006	F02	M014	10	2020-12-23	2021-01-06
8	B005	E01	M017	0	2020-12-27	2021-01-10
9	B004	D01	M010	0	2020-12-25	2021-01-08
10	B002	B01	M002	10	2020-12-18	2021-01-01

Query executed successfully.

III. Execute the select stored procedure order by Member_ID with ASC order.

```

SELECT TOP 10 [IssueNo],
       [B_ISBN],
       [CopyNo],
       [Member_ID],
       [Fine] AS [Fine (Rs.)],
       [IssueDate],
       [ReturnDate]
  FROM [IssueInfo]
 ORDER BY [Member_ID] ASC

```

The screenshot shows the results of the query in a table:

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	B001	A01	M001	10	2020-12-18	2021-01-02
2	B002	B01	M002	10	2020-12-18	2021-01-01
3	B009	I01	M004	0	2020-12-31	2021-01-14
4	B013	M01	M005	0	2021-01-01	2021-01-15
5	B008	H01	M006	0	2020-12-23	2021-01-06
6	B011	K01	M008	0	2021-01-01	2021-01-15
7	B004	D01	M010	0	2020-12-25	2021-01-08
8	B006	F02	M014	10	2020-12-23	2021-01-06
9	B005	E01	M017	0	2020-12-27	2021-01-10
10	B010	J01	M019	0	2020-12-24	2021-01-07

Query executed successfully.

IV. Execute the select stored procedure order by IssueDate with DESC order.

```

SELECT TOP 10 [IssueNo],
       [B_ISBN],
       [CopyNo],
       [Member_ID],
       [Fine] AS [Fine (Rs.)],
       [IssueDate],
       [ReturnDate]
  FROM [IssueInfo]
 ORDER BY [IssueDate] DESC

```

The screenshot shows the results of the query in a table:

IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	B013	M01	M005	0	2021-01-01	2021-01-15
2	B011	K01	M008	0	2021-01-01	2021-01-15
3	B009	I01	M004	0	2020-12-31	2021-01-14
4	B007	G02	M020	0	2020-12-27	2021-01-10
5	B005	E01	M017	0	2020-12-27	2021-01-10
6	B004	D01	M010	0	2020-12-25	2021-01-08
7	B010	J01	M019	0	2020-12-24	2021-01-07
8	B006	F02	M014	10	2020-12-23	2021-01-06
9	B008	H01	M006	0	2020-12-23	2021-01-06
10	B002	B01	M002	10	2020-12-18	2021-01-01

Query executed successfully.

V. Execute the select stored procedure order by ReturnDate with ASC order.

```

SELECT TOP 10 [IssueNo],
[B_ISBN],
[CopyNo],
[Member_ID],
[Fine] AS [Fine (Rs.)],
[IssueDate],
[ReturnDate]
FROM [IssueInfo]
ORDER BY [ReturnDate] ASC
END
EXEC [dbo].[SelectStoredProcedureIssueInfo]

```

2. Update Stored Procedure.

➤ Create and execute the update stored procedure.

```

/*UPDATE STORED PROCEDURE*/
CREATE PROCEDURE UpdateStoredProcedureIssueInfo
AS
BEGIN
SET NOCOUNT ON;
UPDATE [IssueInfo] SET [IssueDate] = '2020-12-17'
WHERE [CopyNo] = 'A01';
END
SELECT [IssueNo],
[B_ISBN],
[CopyNo],
[Member_ID],
[Fine] AS [Fine (Rs.)],
[IssueDate],
[ReturnDate]
FROM [IssueInfo]
ORDER BY [Member_ID] ASC

```

5) STAFF TABLE

1. Create a procedure list of book keepers.

Now, I created another interesting feature that able to view whole information by any single attribute or multiple attributes. For example like this below, I expect to extract only details from book keepers in library,

```
CREATE PROCEDURE listbookkeepers
AS SELECT * FROM StaffTable
WHERE StaffRole = 'Bookkeeper'
GO
```

With that to execute with following command,

```
EXEC listbookkeepers
```

And the result is,

The screenshot shows a SQL query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays a table with four rows of data. The columns are labeled: FirstName, LastName, Contact, Email, StaffID, StaffRole, and Joined_Date. The data is as follows:

	FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date
1	Dasitha	Samarasinghe	772899112	dsamare21@gmail.com	1	Bookkeeper	2002-01-19
2	Chamodh	Jayasekara	772569112	chamo41@gmail.com	3	Bookkeeper	2005-08-23
3	Kevin	Koththigoda	768392313	kevinkoththi90@gmail.com	6	Bookkeeper	2009-01-02
4	Dhanuska	Samarasinghe	772892312	danusamare12@gmail.com	7	Bookkeeper	2009-09-18

2. Create a procedure for admin info with credentials

I used staff id here because it helps to view whole staff members' info those who have credentials but this time with the credentials too. To this I used create procedure function as it joins both StaffTable and AuthSys table,

```
~  
CREATE PROCEDURE admindetailswithcredentials  
AS  
SELECT * FROM StaffTable AS S  
JOIN AuthSysTable AS A  
ON S.StaffID = A.StaffID  
GO
```

With this, result has to be executed by using following,

```
EXEC admindetailswithcredentials
```

The result can be like this,

	FirstName	LastName	Contact	Email	StaffID	StaffRole	Joined_Date	StaffID	Username	Password
1	Vindya	Ravindi	732499112	vini45@yahoo.com	11	Admin	2013-02-24	11	Nethi	abc123

6) PUBLISHER TABLE

1. Procedure to find the publisher by ID

The screenshot shows the SQL Server Management Studio (SSMS) interface. In the top pane, a script window contains the following T-SQL code:

```
CREATE PROCEDURE find_publisher
    @id varchar (5)
AS
    SELECT Pub_ID , Pub_Name , Pub_Contact
    FROM Publisher
    WHERE Pub_ID = @id

    EXEC find_publisher '1002';
    EXEC find_publisher '1009';
```

In the bottom pane, there are two result sets. The first result set, titled "Results", shows the output of the first execution of the procedure:

Pub_ID	Pub_Name	Pub_Contact
1002	Macmillans	0557484871

The second result set, also titled "Results", shows the output of the second execution of the procedure:

Pub_ID	Pub_Name	Pub_Contact
1009	Archid constable and company	0769874471

- 7) I have created a procedure to find the publisher. The above is the result after creating and executing the procedure.
- 8) In here, by searching the Pub_ID we can get information about publisher name and the publisher contact.

2. Function to find the number of publishers

The screenshot shows the SQL Server Management Studio (SSMS) interface. In the top pane, a script window contains the following T-SQL code:

```
CREATE PROCEDURE noOfPublishers
@Pub_ID varchar (5)
AS
BEGIN
    DECLARE @totpublishers int
    SELECT @totpublishers = count(Pub_ID) FROM Publisher
    RETURN @totpublishers
END

DECLARE @pubValue int
EXEC @pubValue = noOfPublishers ''
PRINT @pubValue
```

In the bottom pane, the "Messages" tab displays the output of the function execution:

15
Completion time: 2021-01-03T20:05:03.0176716+05:30

- This function is the result after creating and executing the function.
- By using this function, we can find out how many publisher records are there in total.

4. SECTION 04

❖ CRITICAL EVALUATION

As per the scenario we were required to create a SQL for a library management system. This system provides and have many functions that helps the library network to work easily. There are many parties involved in the library system. Analysis of events like tracking resources like books, copies can be monitored via this implementation. The main functions of this library system are,

- It provides user / borrower to buy books on either cash or by loan.
- There is a separate system which records the details of the borrowers and they can update, delete, and insert new or existing users to the library system.
- Also, the details of the books are too stored in the library system whereas the borrowing and the returning date will also be recorded. And there is a fine been charged for the late returning of the books.
- Moreover, it also stores details about the publishers and the other branches of the library.
- Staff details are also being recorded in the system and any updates and deletions of the staff can also be made.

However as everything has its good and bad both, this library system too has its own drawbacks.

- Our system is not capable of online payments so that people can make their purchases more easily.
- This would be not strong enough to be immune to activities from unauthorized parties like injection, hacking and sniffing activities.
- Accessing system can be sometime depends on network, as more and more computers connect to the network, it can reduce the speed of accessing fairly until the modern technology provide a solution for it.

❖ **FUTURE IMPLEMENTATION**

- When it comes to the future implementations keeping a step forward from our existing library system, we can create an online payment method to the system.
- We are planning to make it more user friendly and appealing and attractive.
- We are planning to get more parties involved to the system and make it possible for everyone to satisfy their desires in one roof when it comes with books.
- We also plan to develop a platform where users can come online read books while making payments online.
- We are planning on providing devices that would be more user friendly to access our library services anywhere for our customers.