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

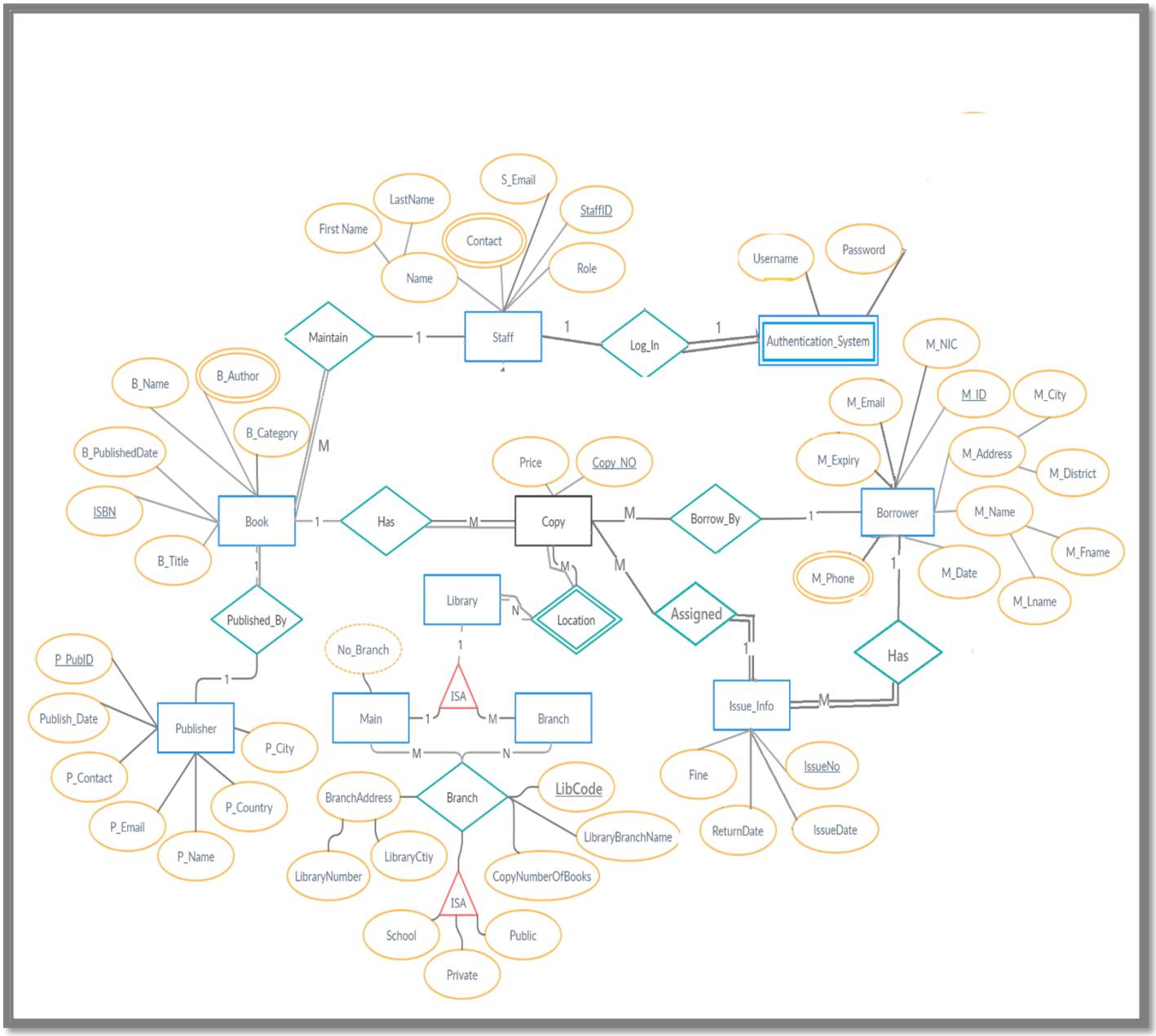
Table of Contents

1. SECTION 01

❖ 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

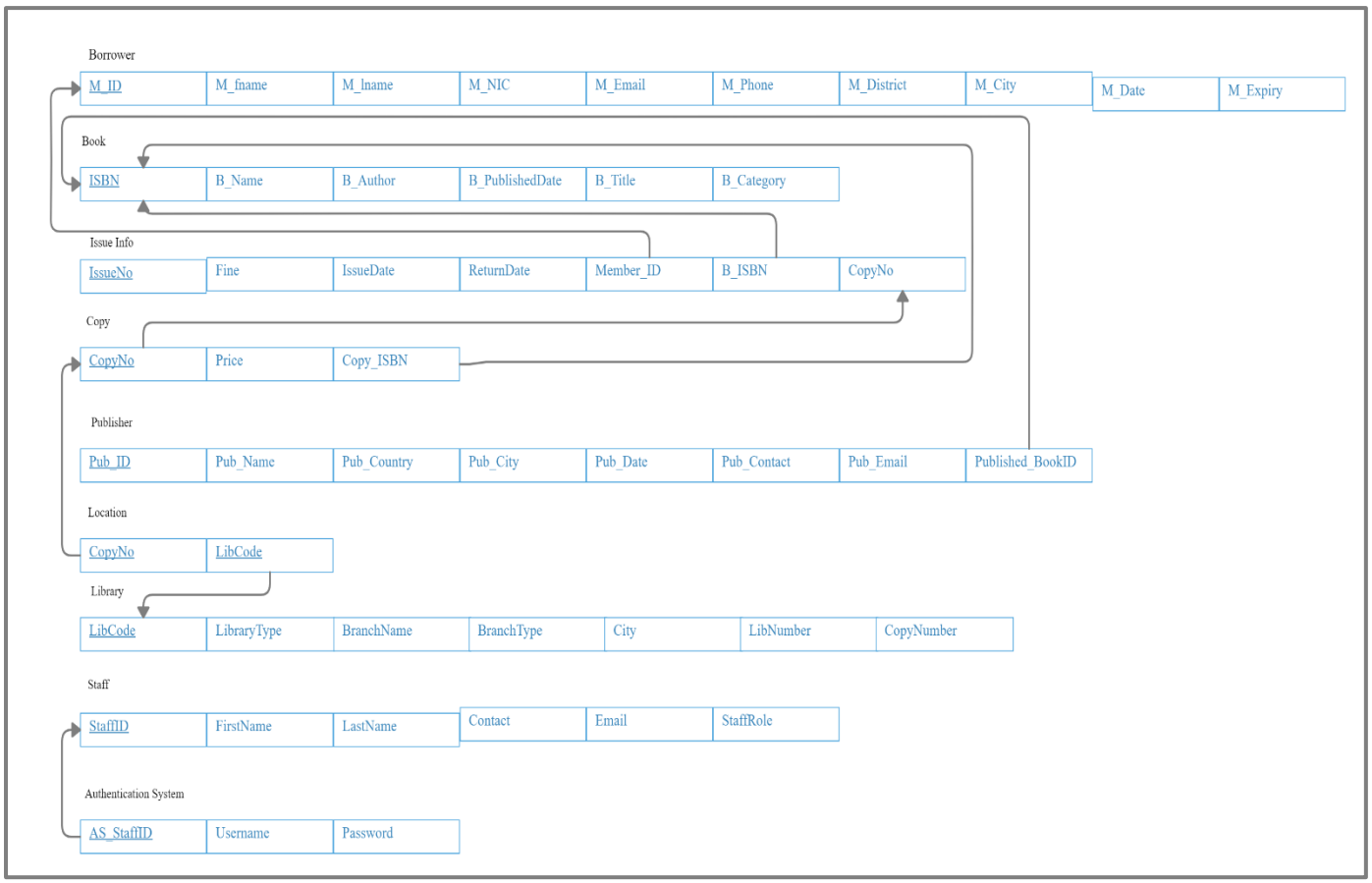


❖ ASSUMPTIONS MADE REGARDING THE EERD

- 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 recieved 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 adiitional 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.

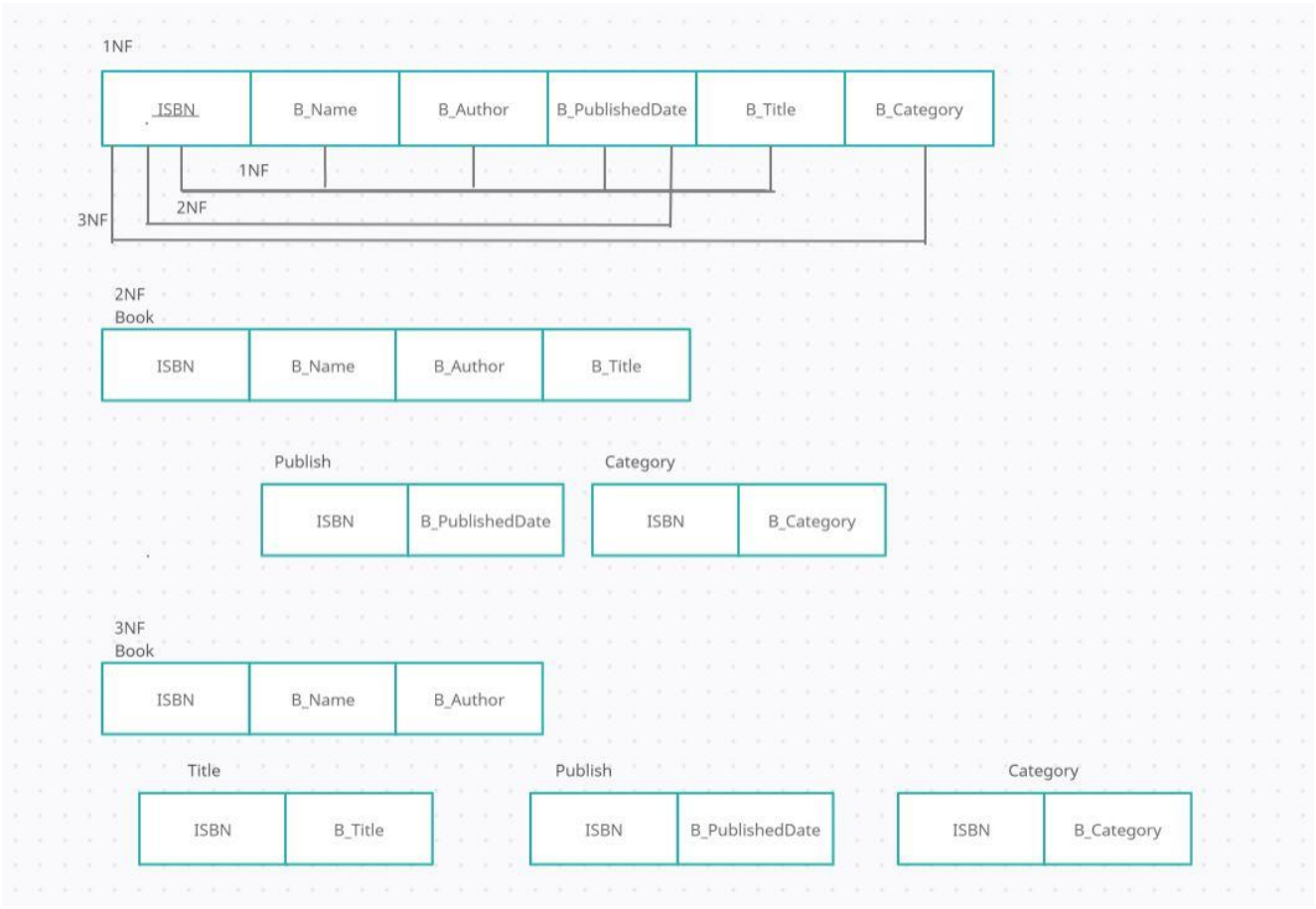
- Staff members who want log into the system are required to provide the valid username and password. Using username and password can prevent unauthorized parties to login the system.
- Staff members who have thier role set to admin are considered as administrators and only them can login the system.
- 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

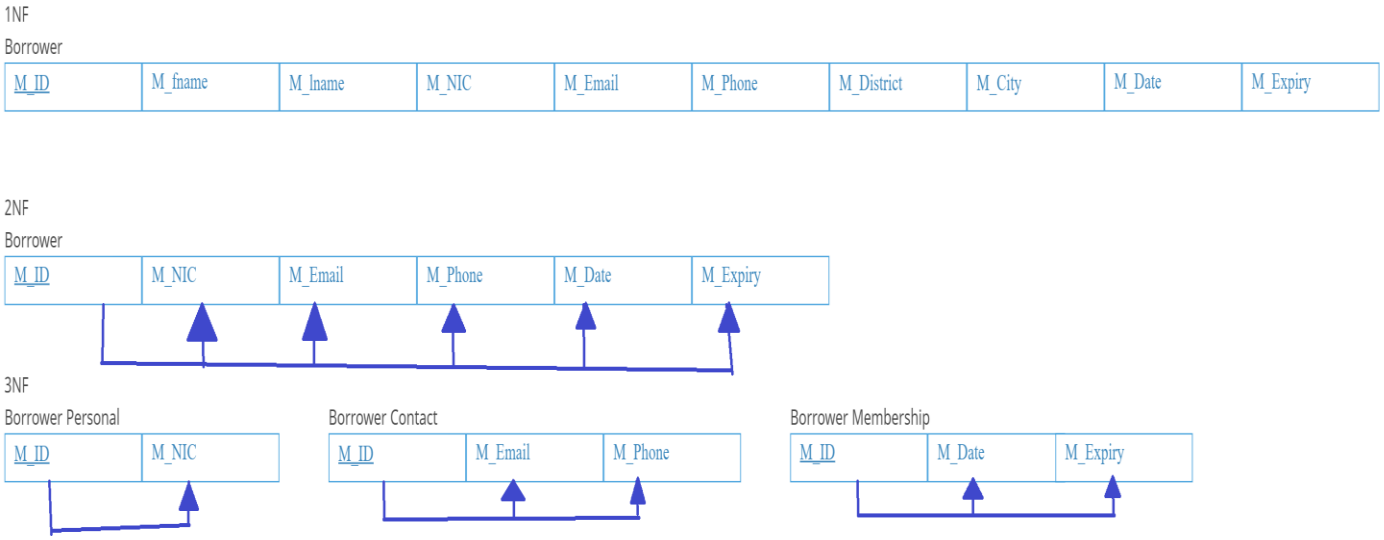


❖ NORMALIZATION

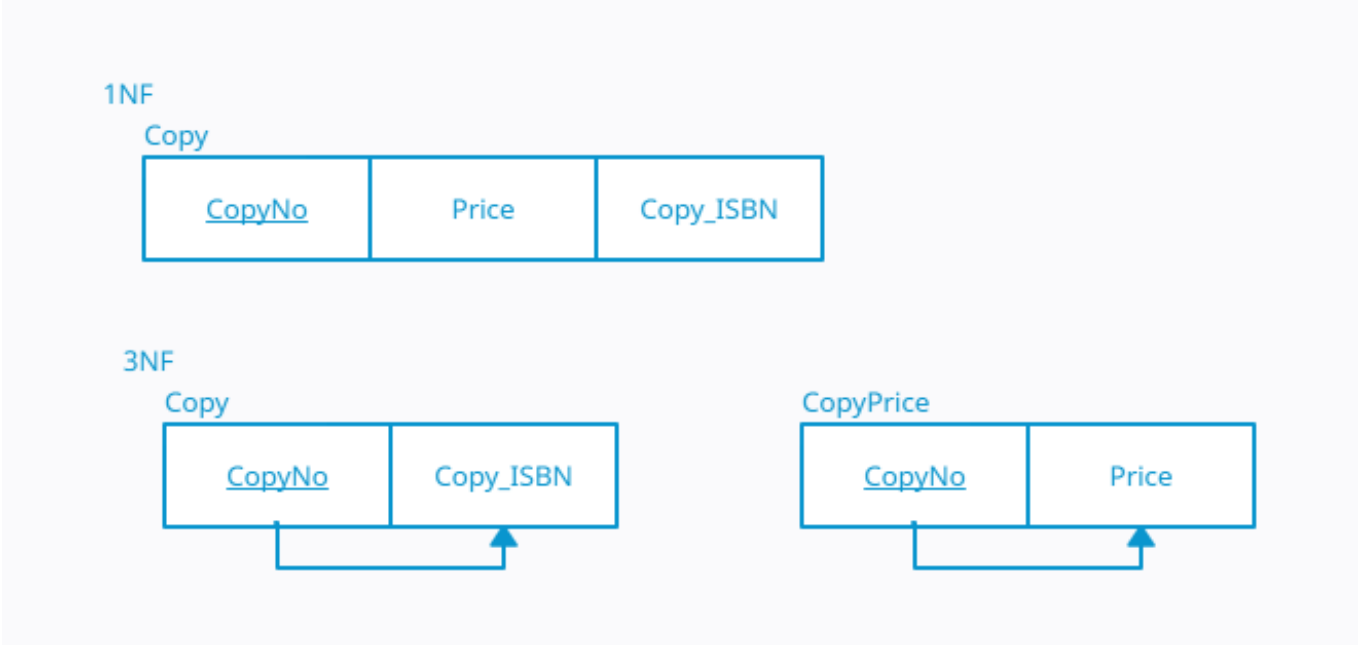
1) BOOK TABLE



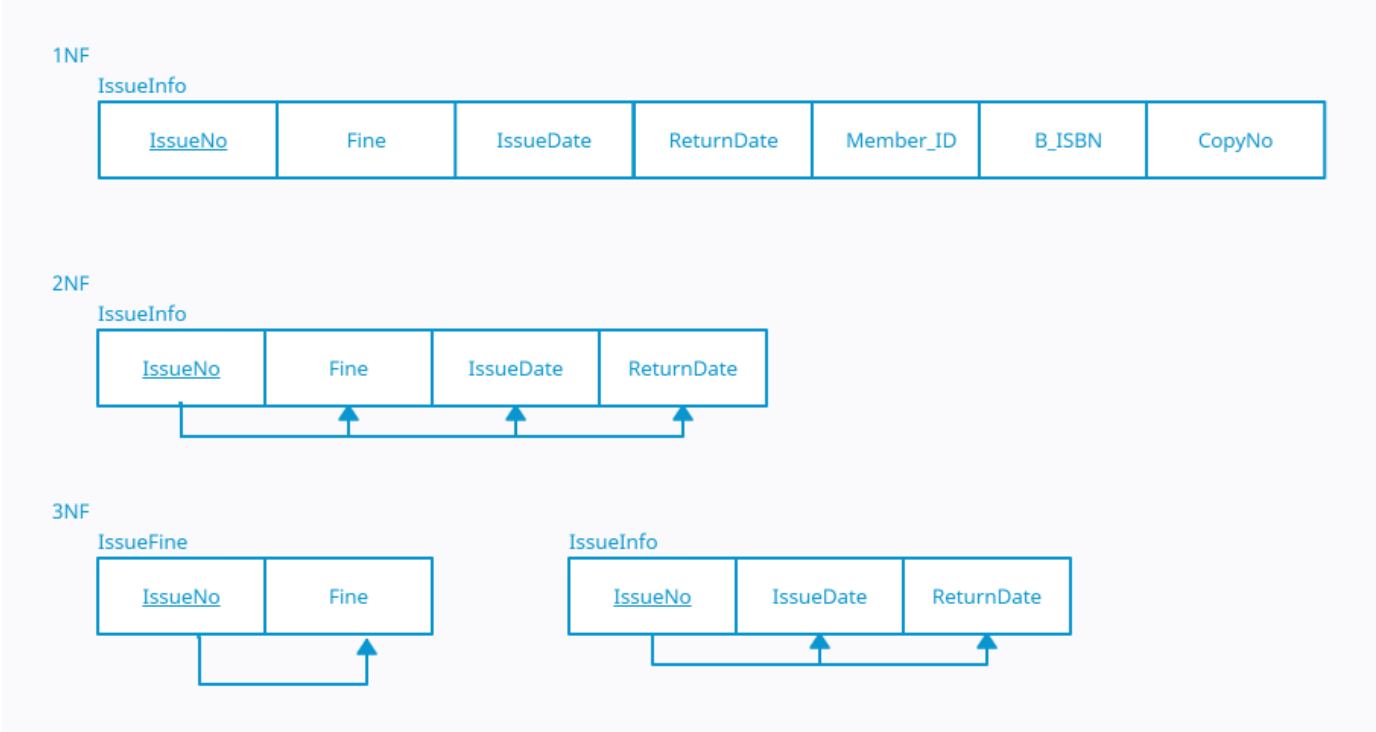
2) BORROWER TABLE



3) COPY TABLE



4) ISSUEINFO TABLE



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)

StaffID	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

StaffID	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

StaffID	FirstName	LastName	Contact	Email	Joined Date	Staff Role

Authentication System Table

Staff ID - FK	User Name	Password

6) LIBRARY TABLE

Library

1nf

libcode	LibraryType	BranchName	BranchType	city	LibNumber	copynumber

2nf

libcode	BranchName	city	LibNumber

3nf Name details

libcode	BranchName

address

libcode	City	libNumber

7) PUBLISHER TABLE

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

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.

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.

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.

❖ 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 should 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

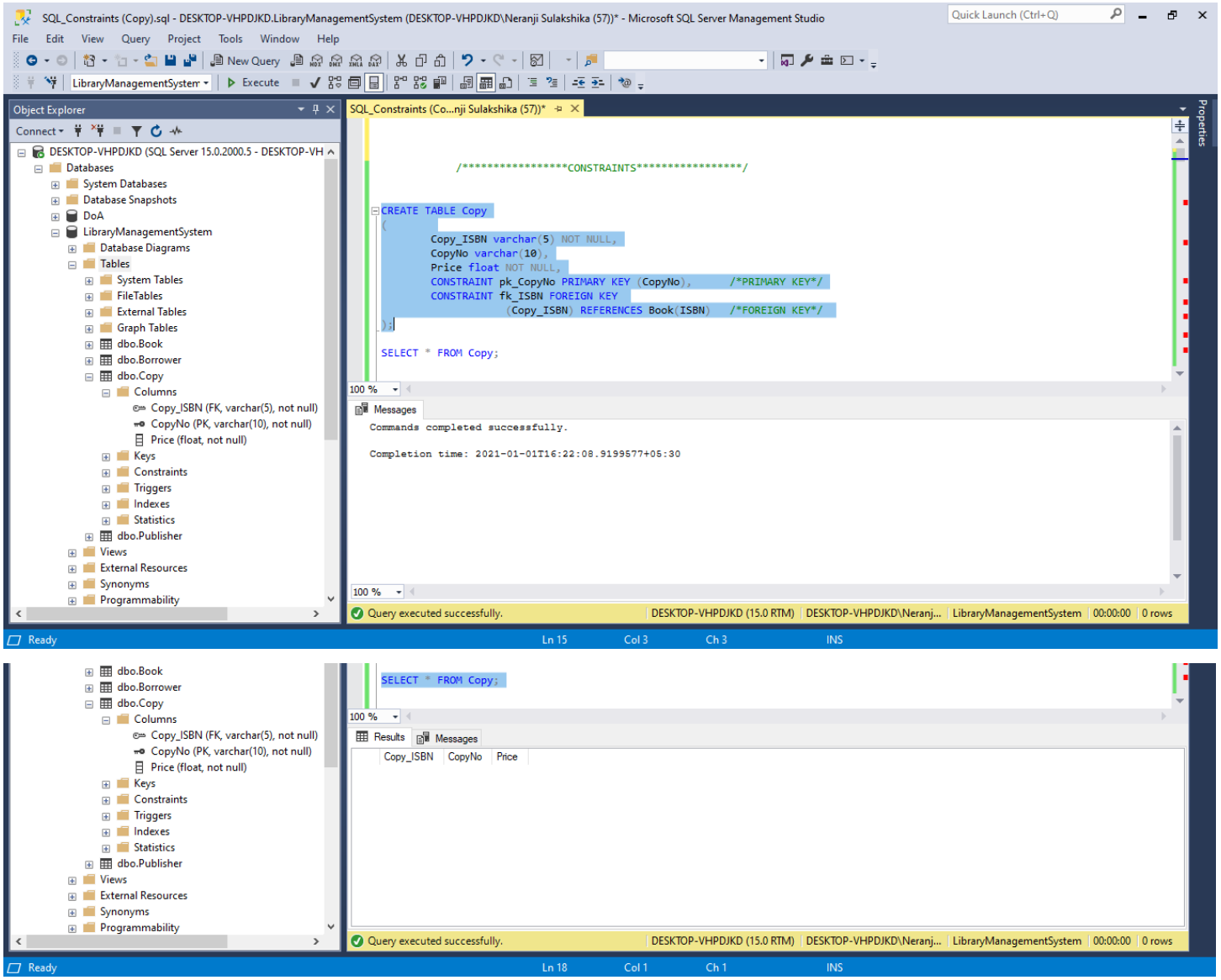
CONSTRAINTS FOR BORROWER TABLE

```
CONSTRAINT pk_mid PRIMARY KEY (M_ID),
CONSTRAINT uq_mnic UNIQUE (M_NIC),
CONSTRAINT uq_memail UNIQUE (M_Email),
CONSTRAINT uq_mphone 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

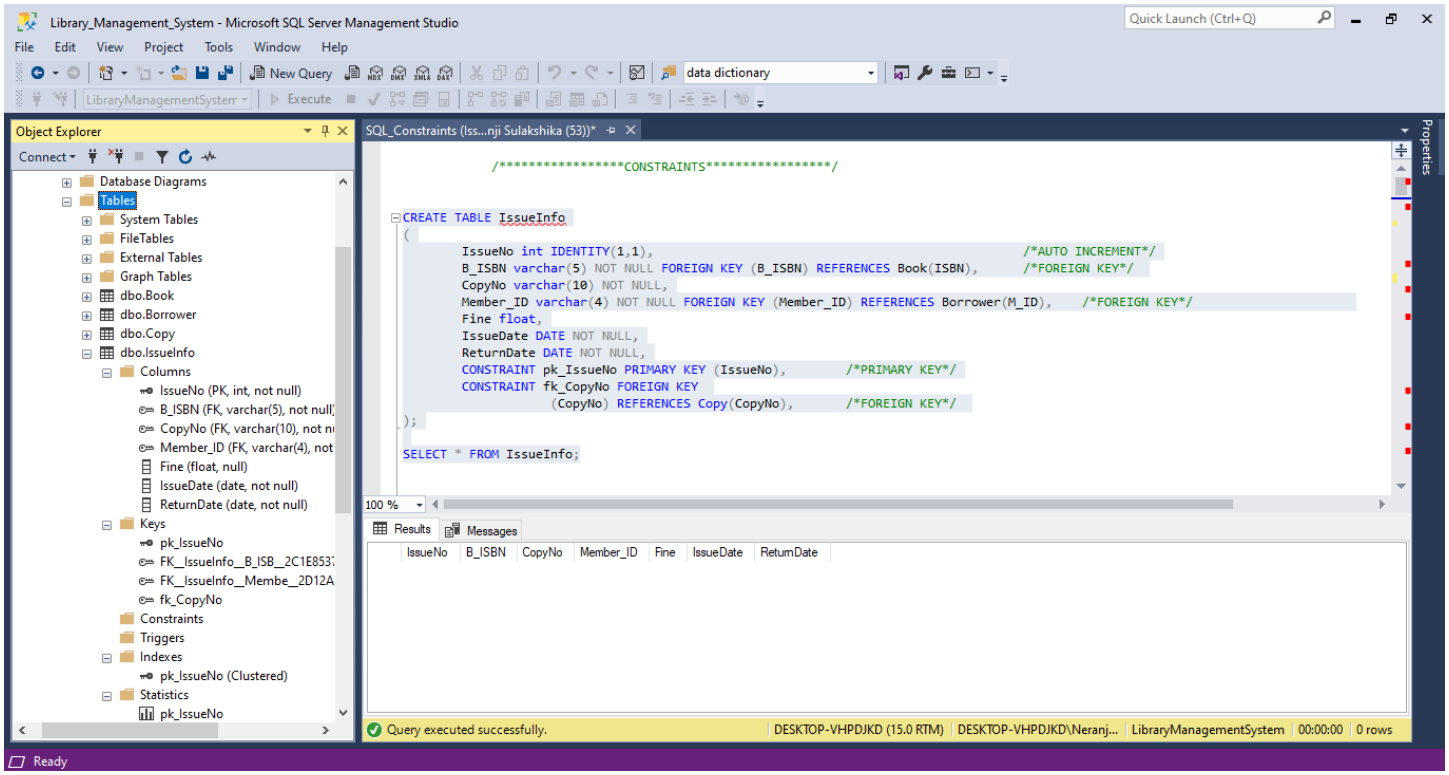
1. Create table and use constraints.



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.



5) STAFF TABLE

Create table Staff

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

```
SQLQuery1.sql - LA...15503\Dasitha (74))*
CREATE DATABASE Staff;
```

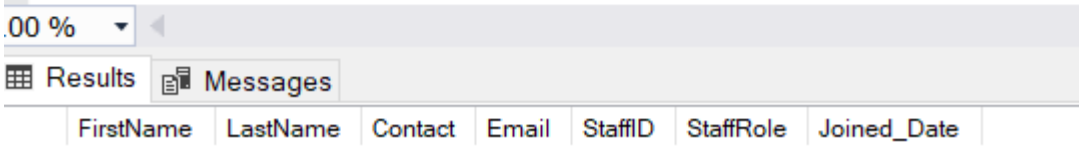
After creating database Staff and USE statement, created a table called StaffTable to insert details of all staff members considering StaffID as a primary key.

```
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
```



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 abled to 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 */
);
```

```
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 */
```

```
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),);
```

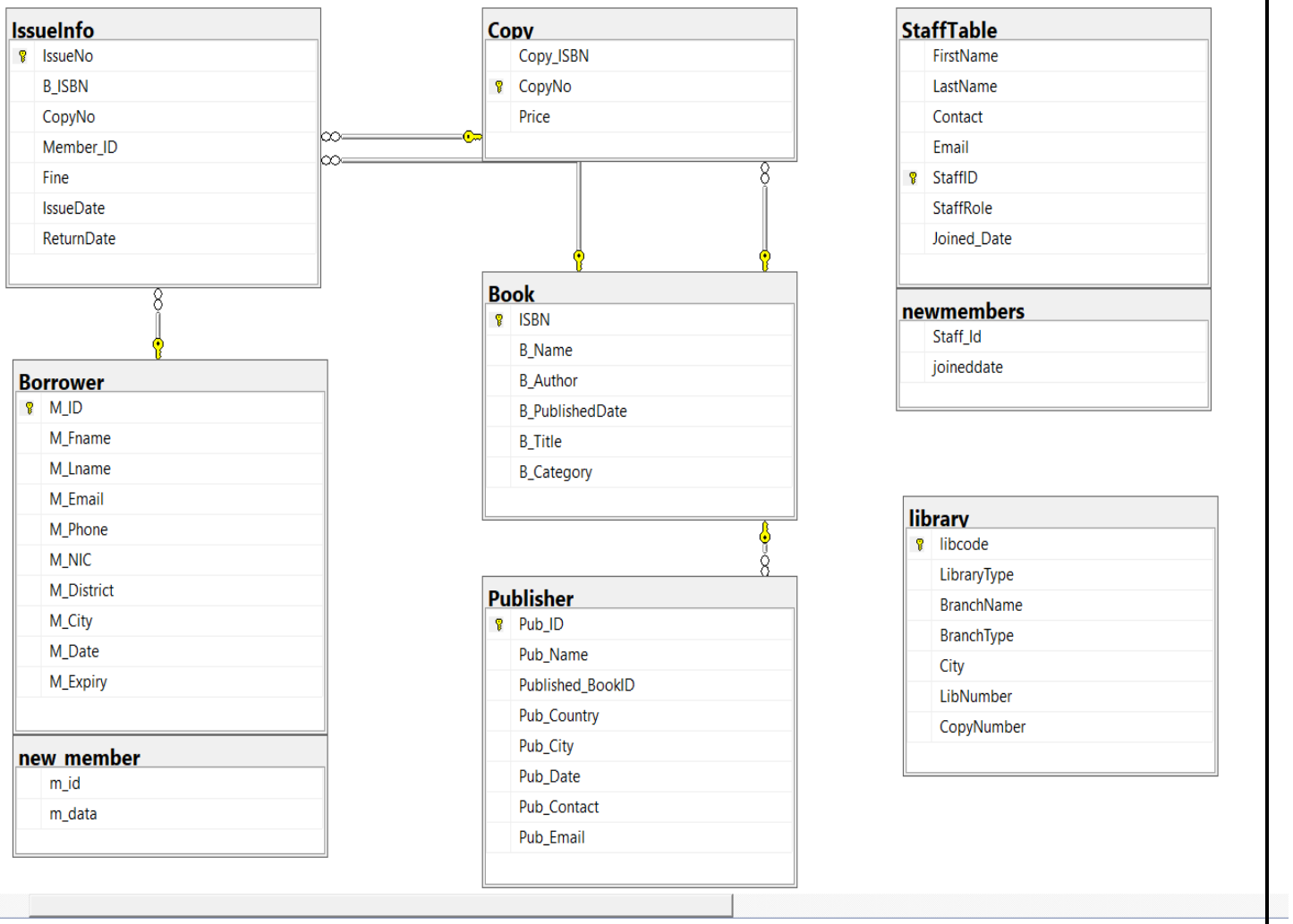
50 %

Results | Messages

	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

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. Apartfromthat the last column (copynumber) stores the number of copies received for each library.

❖ DB DIAGRAMS


IssueInfo
IssueNo
B_ISBN
CopyNo
Member_ID
Fine
IssueDate
ReturnDate
Copy
Copy_ISBN
CopyNo
Price
StaffTable
FirstName
LastName
Contact
Email
StaffID
StaffRole
Joined_Date
newmembers
Staff_Id
joineddate
Book
ISBN
B_Name
B_Author
B_PublishedDate
B_Title
B_Category
Borrower
M_ID
M_Fname
M_Lname
M_Email
M_Phone
M_NIC
M_District
M_City
M_Date
M_Expiry
library
libcode
LibraryType
BranchName
BranchType
City
LibNumber
CopyNumber
Publisher
Pub_ID
Pub_Name
Published_BookID
Pub_Country
Pub_City
Pub_Date
Pub_Contact
Pub_Email
new_member
m_id
m_data

❖ SAMPLE RECORDS

1) BOOK TABLE

DATA INSERTION TO THE TABLE

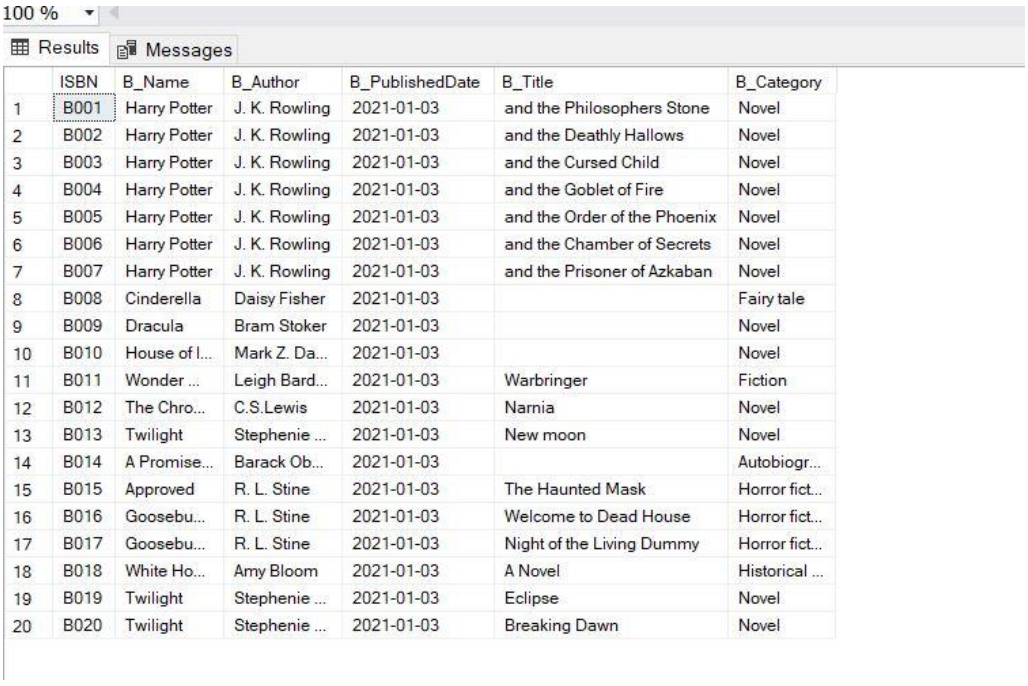
Data insertion are shown below with the results.

```
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 %

Results | Messages

	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','Gurusingha','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

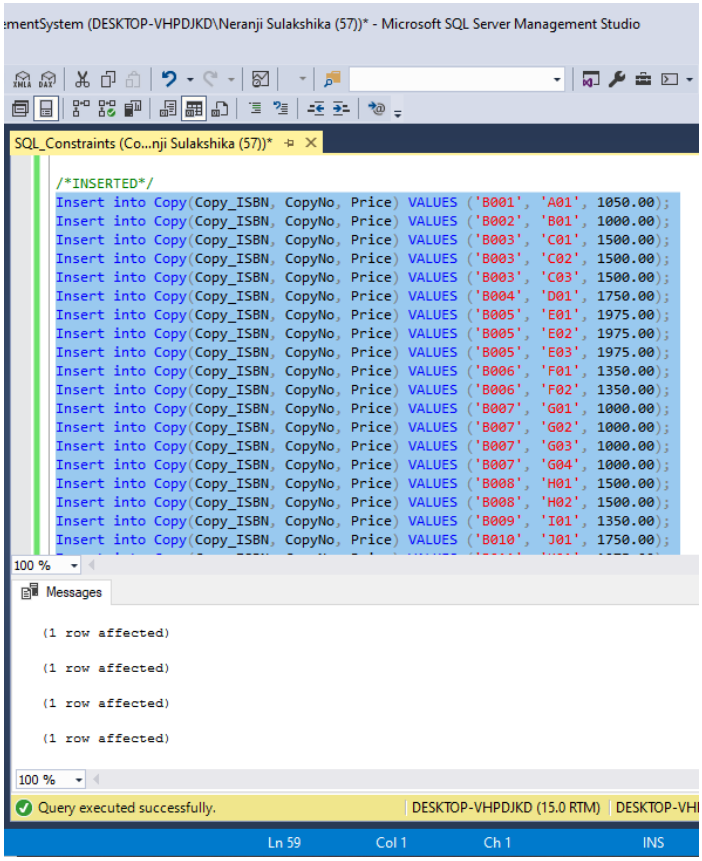
Results | Messages

	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	Gurusingha	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	pusthakaalaya@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	abrakadabra@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

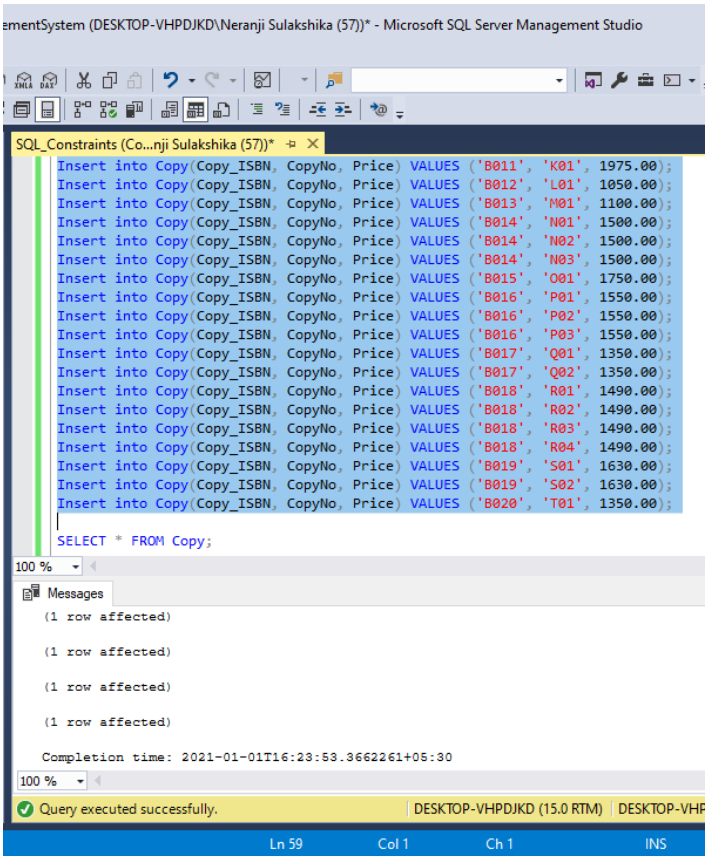
1. Insert sample data



```
/*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);
```

```
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
```

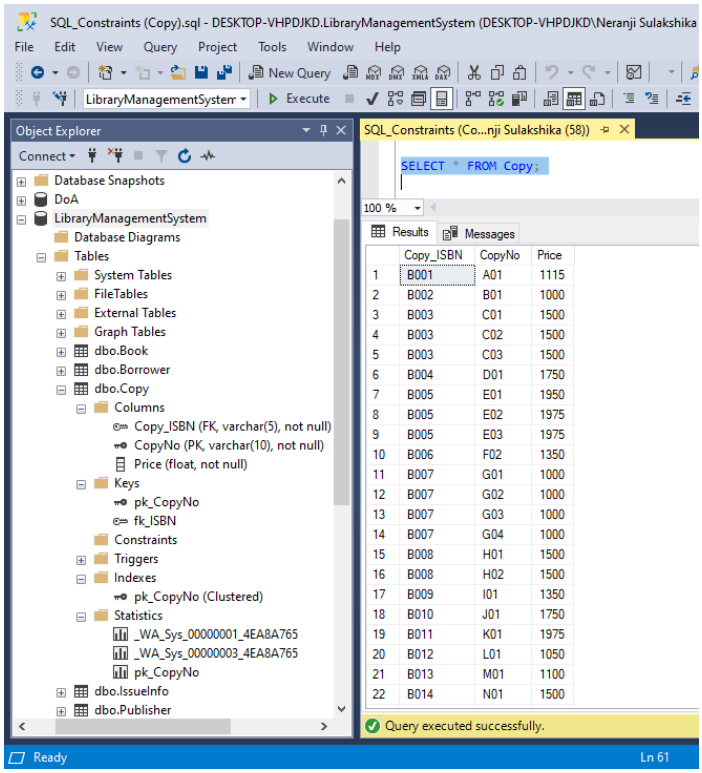
Query executed successfully.



```
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 ('B018', 'R04', 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;
```

```
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
Completion time: 2021-01-01T16:23:53.3662261+05:30
```

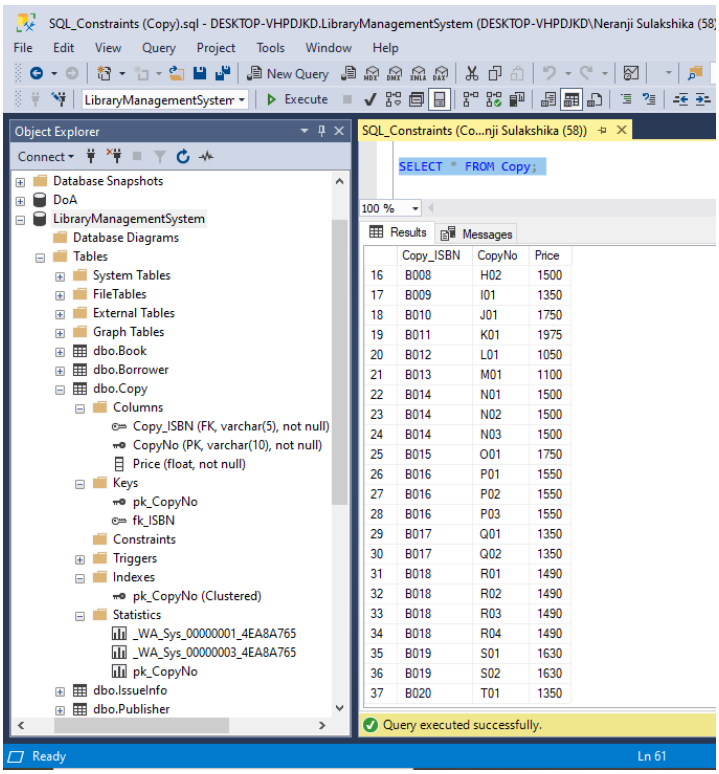
Query executed successfully.



```
SELECT * FROM Copy;
```

	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

Query executed successfully.



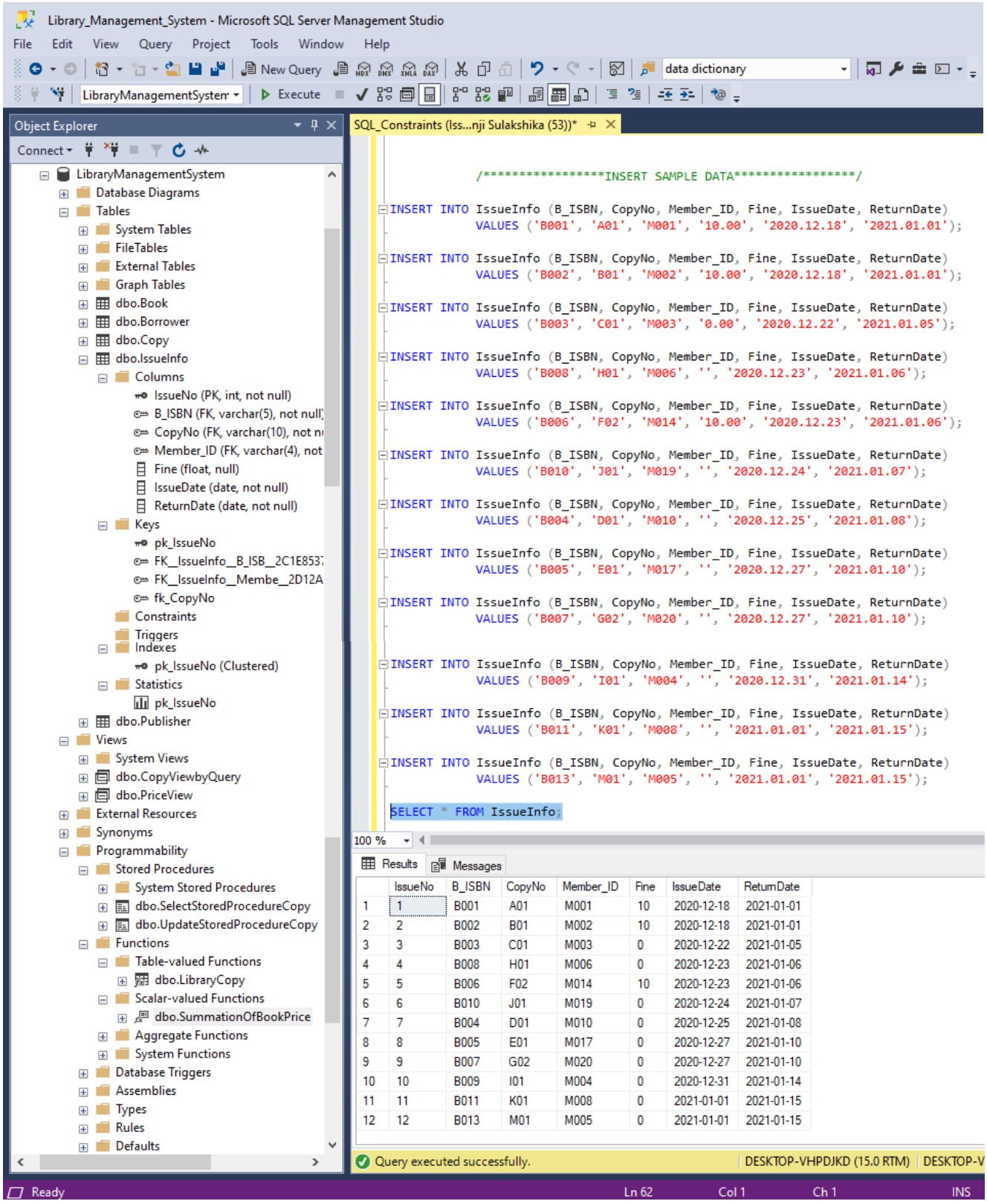
```
SELECT * FROM Copy;
```

	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

Query executed successfully.

4) ISSUE-INFO TABLE

1. Insert sample data



5) STAFF TABLE

Insert table staff

Started inserting data(s) into the Staff Table,

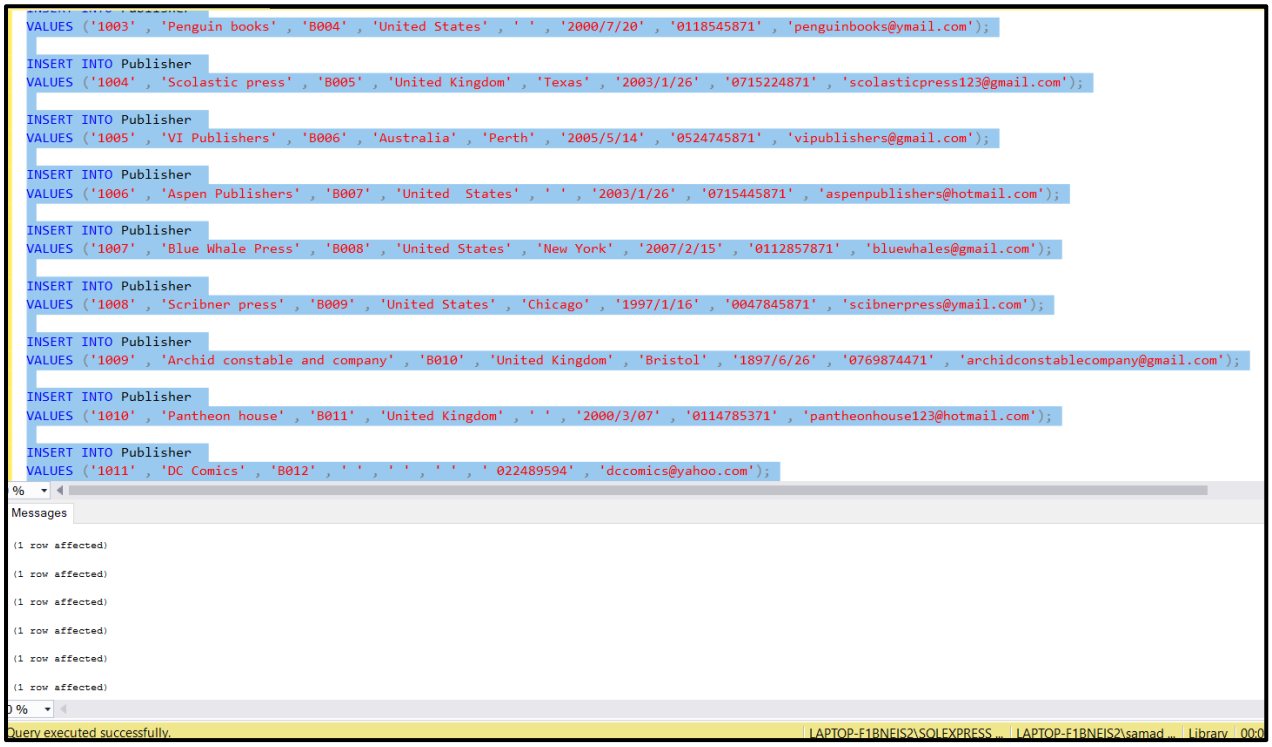
```
INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Dasitha','Samarasinghe','0772899112','dsamare21@gmail.com','Bookkeeper','2002-01-19')
INSERT INTO StaffTable (FirstName,LastName,Contact,Email,StaffRole,Joined_Date) VALUES ('Shevon','Marasinghe','0772431132','shemara67@gmail.com','Cleaner','2005-03-15');
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-
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,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','rohithagune32@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	rohithagune32@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' , 'scibnerpress@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)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
```

Query executed successfully.

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	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	scibnerpress@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			1900-01-01	022489594	dccomics@yahoo.com
13	1012	Simon & Shuster	B013	United Kingdom	Cardiff	2014-04-15	0112478871	simon&shutter2gmail.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' ,'3
insert into library values ('010' , 'branch','oxford','public','Thalawathugoda', '04/45' ,'27'
```

50 %

Results | Messages

	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.

```
/*************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;
```

```
.00 %
Messages
Commands completed successfully.

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 = b.ISBN
      AND i.B_Name = 'Approved'
 END
 GO

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

 SELECT * FROM Book
```

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 l...	Mark Z. Da...	2021-01-03		Novel
B011	Wonder ...	Leigh Bard...	2021-01-03	Warbringer	Fiction
B012	The Chro...	C.S.Lewis	2021-01-03	Narnia	Novel
B013	Twilight	Stephenie ...	2021-01-03	New moon	Novel
B014	A Promise...	Barack Ob...	2021-01-03		Autobiogr...
B015	Approved	R. L. Stine	2021-01-03	The Haunted Mask	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;
```

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

```
INSERT INTO Borrower
VALUES ('M007','N','J','god@gmail.com','0706000000','20013650004v','GALLE','Fort','2000-01-01','2020-01-01');
```

100 %

Results | Messages

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	2020-01-0
8	M008	Manoja	Perera	hi@gmail.com	707000000	20015000002v		Negombo	2000-01-01	2020-01-01

	m_id	m_data
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
8	M007	New Member with Membership ID = M007has joined to the library system as at Jan 3 2021 11:59AM

Query executed successfully.

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

```
DELETE FROM Borrower WHERE M_ID = 'M007'
```

100 %

Results | Messages

6	M006	Irusha	Samarawikrama	thomas@gmail.com	705000000	20013650003v	COLOMBO	Maharagama	2000-01-01	2020-01-01
7	M008	Manoja	Perera	hi@gmail.com	707000000	20015000002v		Negombo	2000-01-01	2020-01-01
8	M009	Mendika	Gurusingha	sidecut@gmail.com	708000000	20015000003v	GALLE		2000-01-01	2020-01-01

	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 @m_id = M_ID from inserted
INSERT INTO new_member
VALUES (@m_id, 'Member with Membership ID = ' + CAST (@m_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

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

100 %

Results | Messages

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	NULL	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	Gurusingha	sidecut@gmail.com	708000000	20015000003v	GALLE		2000-01-01	2020-01-01

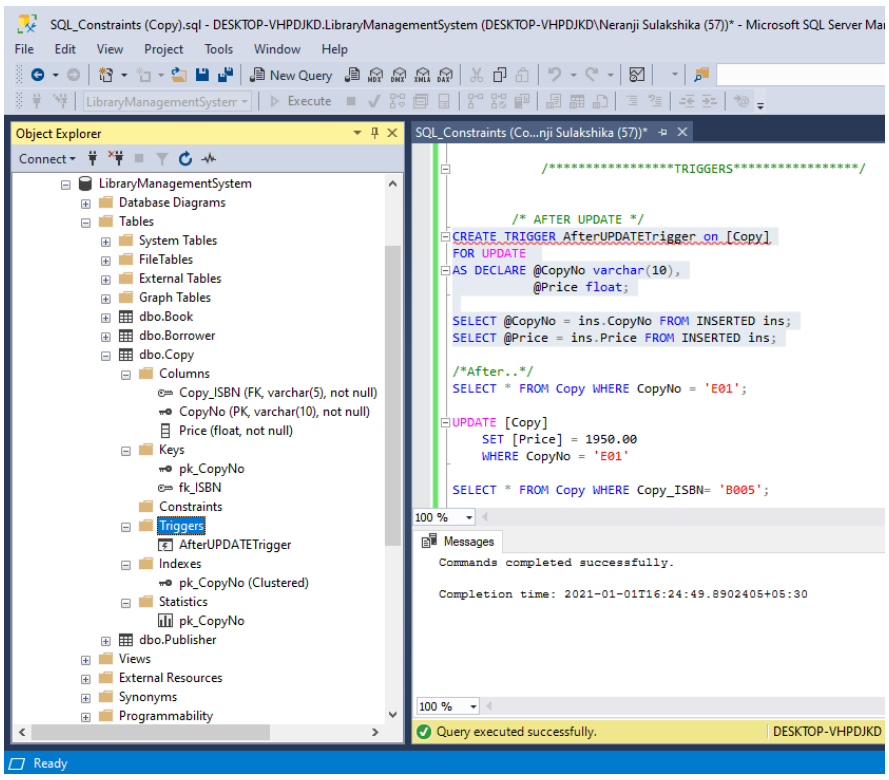
	m_id	m_data
12	M012	New Member with Membership ID = M012has joined to the library system as at Jan 3 2021 11:39AM
13	M013	New Member with Membership ID = M013has joined to the library system as at Jan 3 2021 11:39AM
14	M014	New Member with Membership ID = M014has joined to the library system as at Jan 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

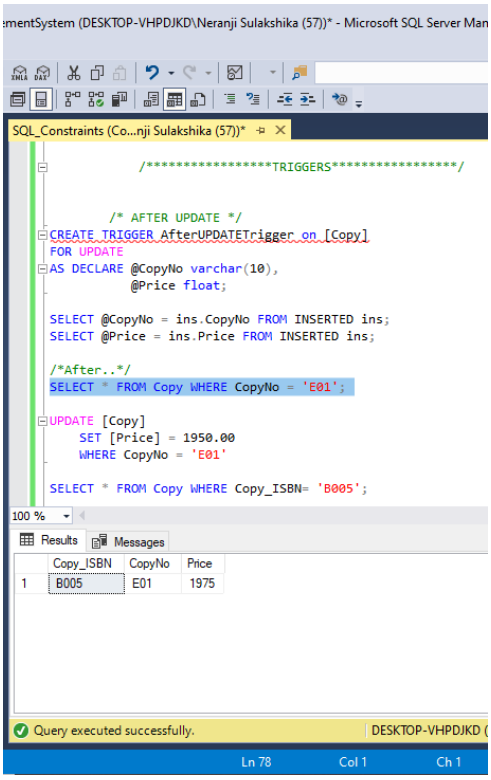
Query executed successfully.

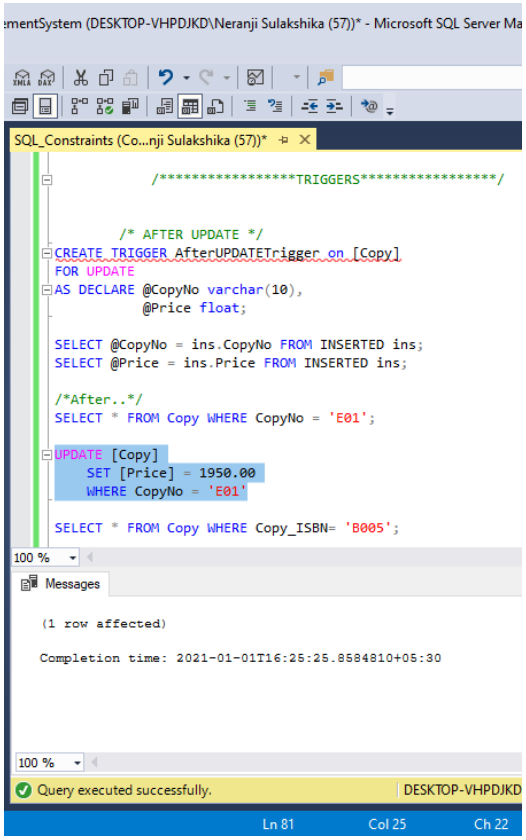
3) COPY TABLE

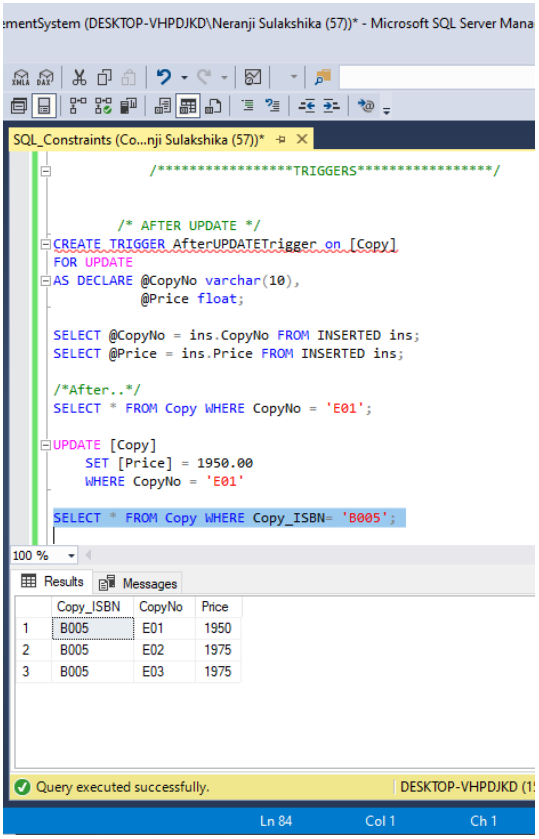
1. After Update Trigger

- Create and execute the ‘AfterUpdateTrigger’.



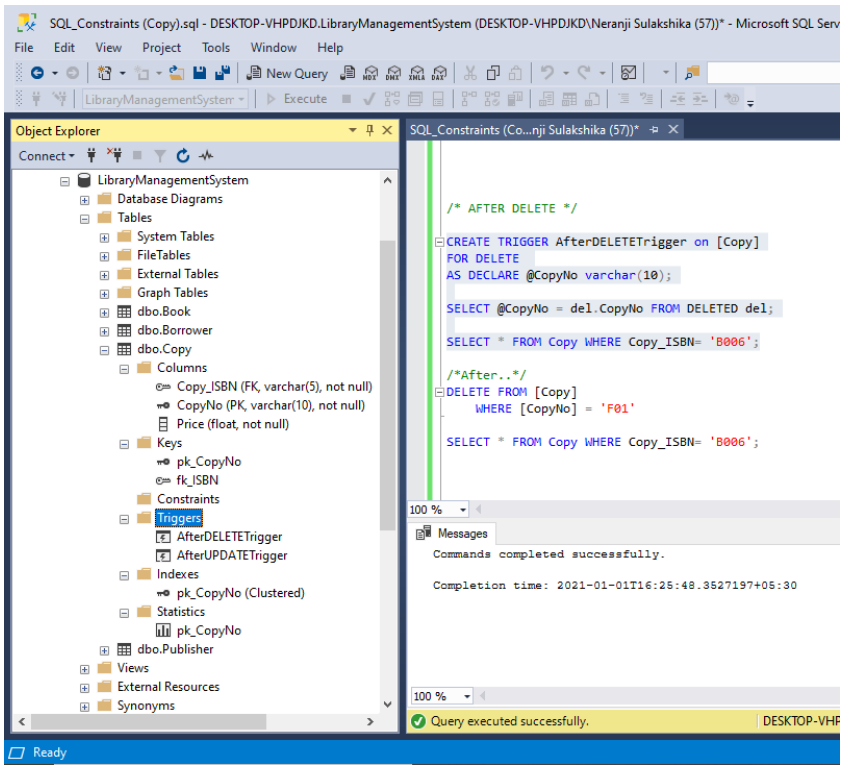


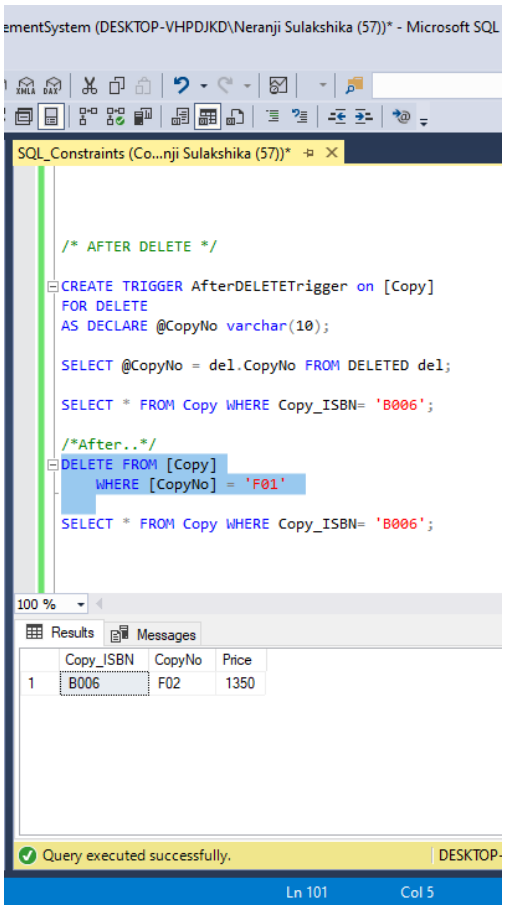


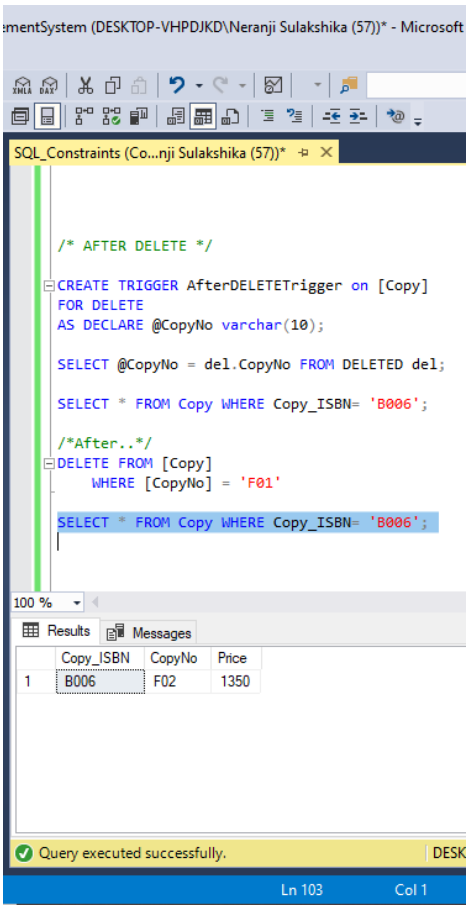


1. After Delete Trigger

- Create and execute the ‘AfterUpdateTrigger’.

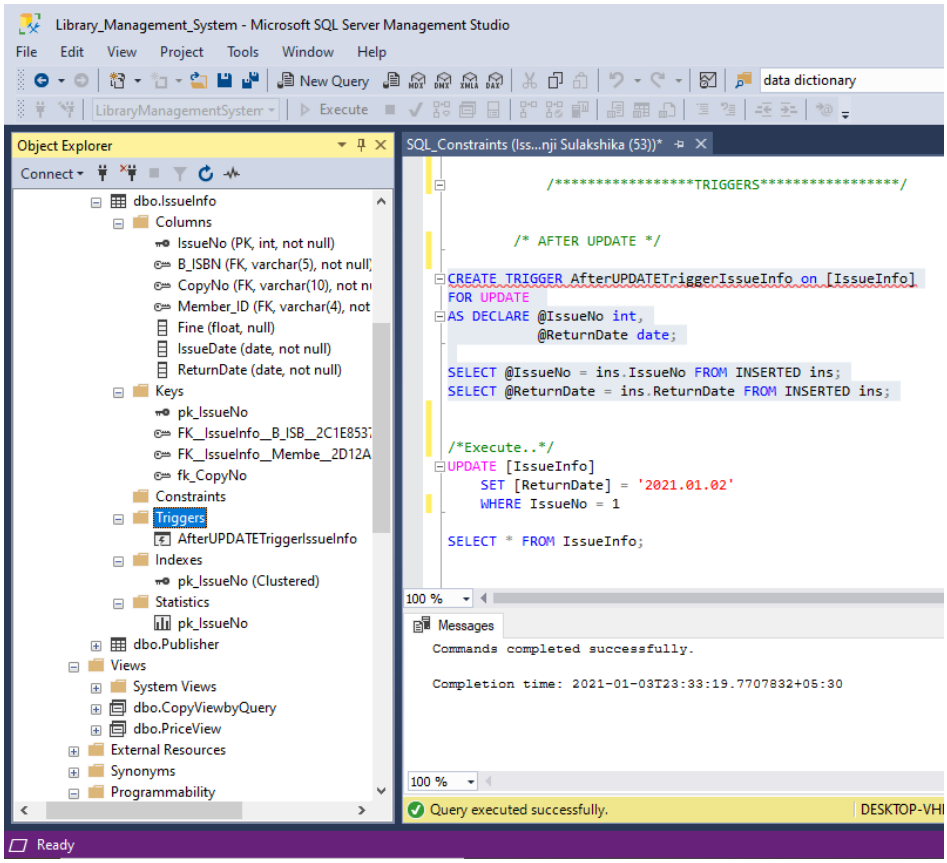




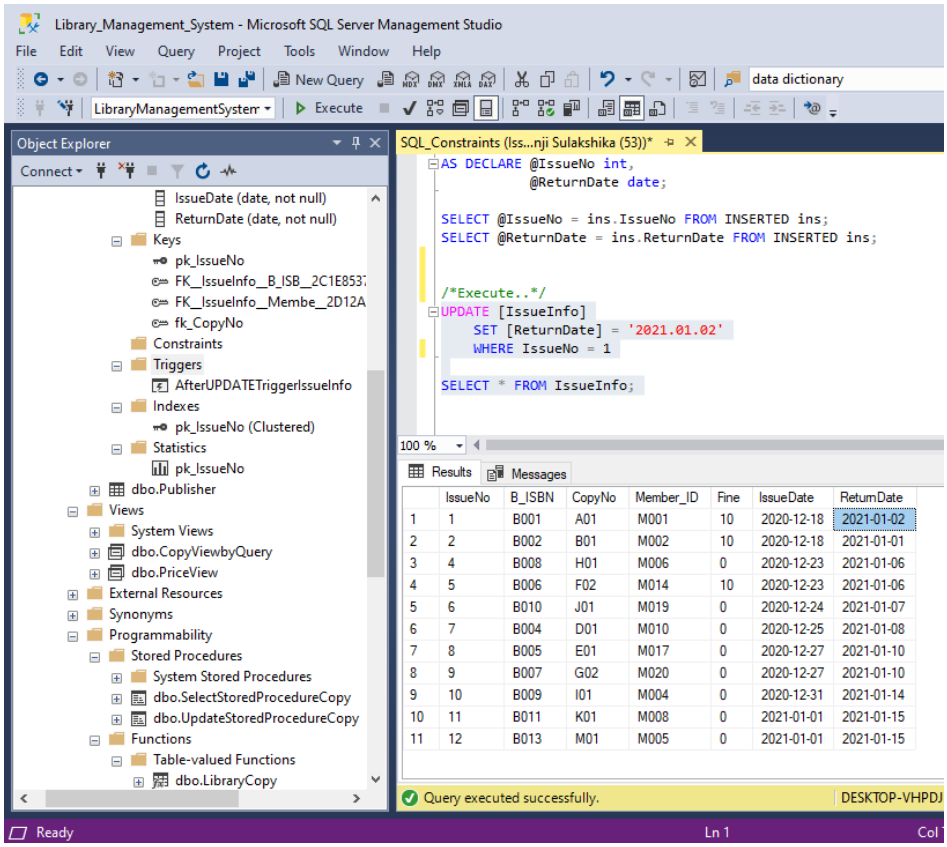


4) ISSUE-INFO TABLE

1. After Update Trigger

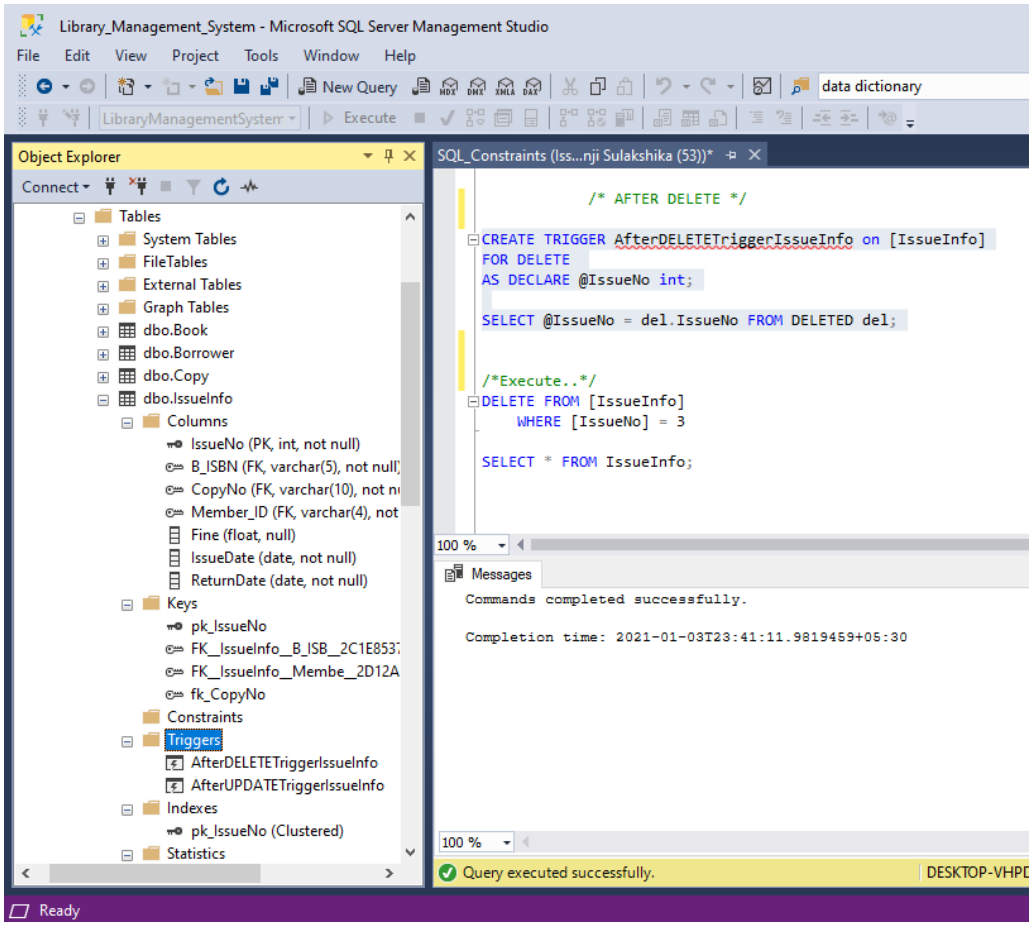


This After Update Trigger is created to update the return date

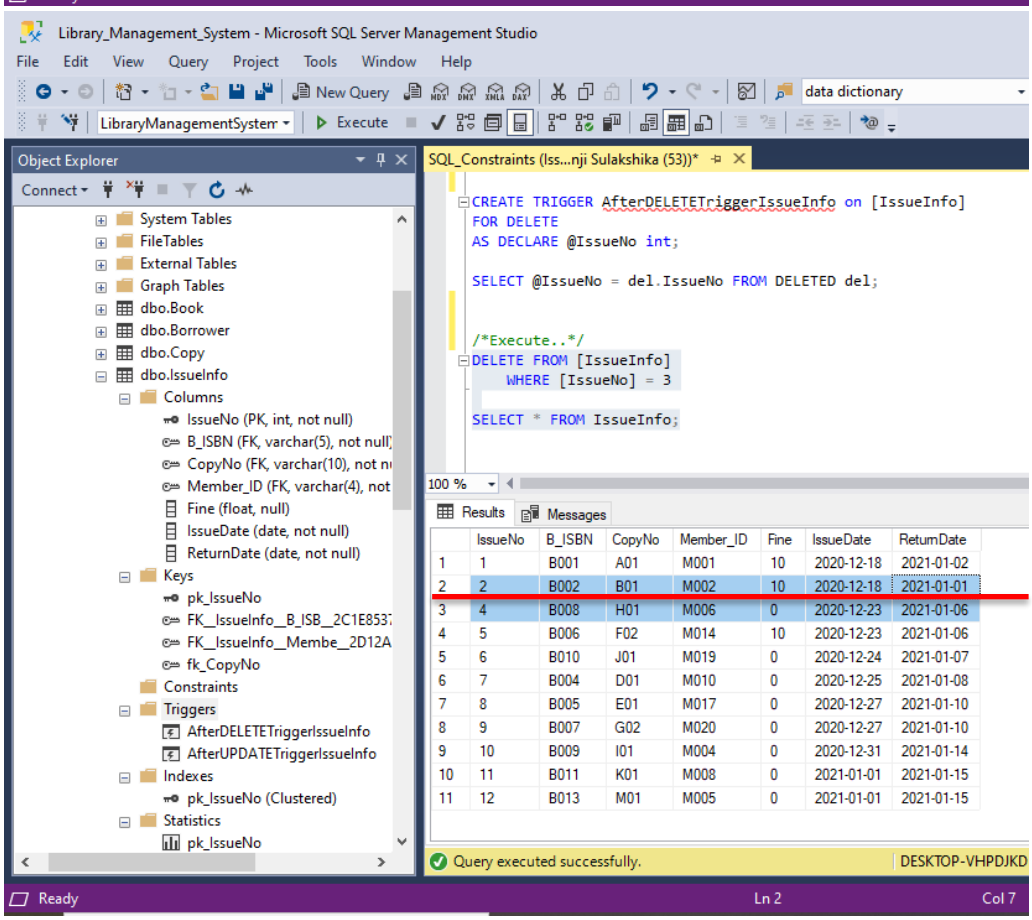


After execut this, we can see ‘IssueNo 1’ related Return date updated to ‘2020.01.02’.

2. After Delete Trigger



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



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

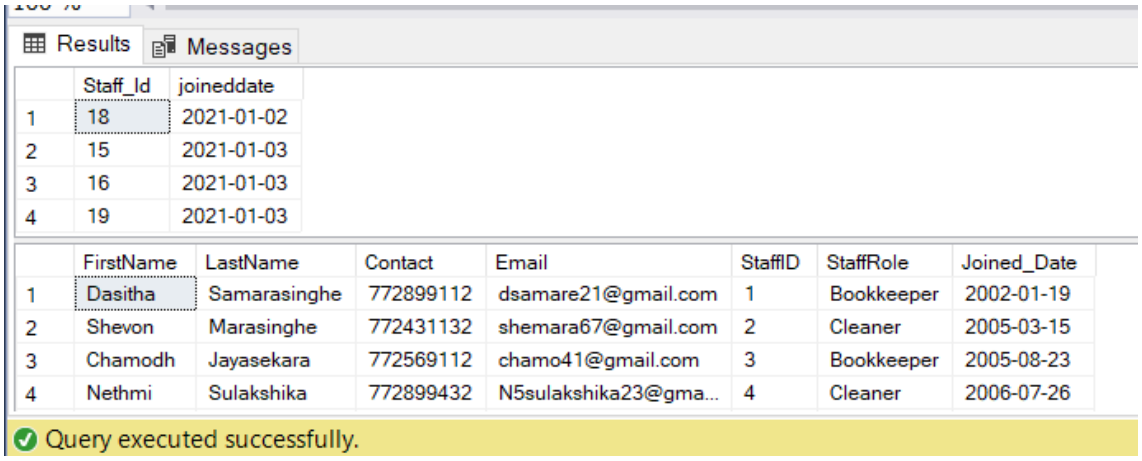
5) STAFF TABLE

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,



Results | Messages

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

	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

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,

```
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
```

0 %

Results | Messages

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	Udana	722499112	NULL	9	Cleaner	2012-03-12

Query executed successfully

After Delete Trigger

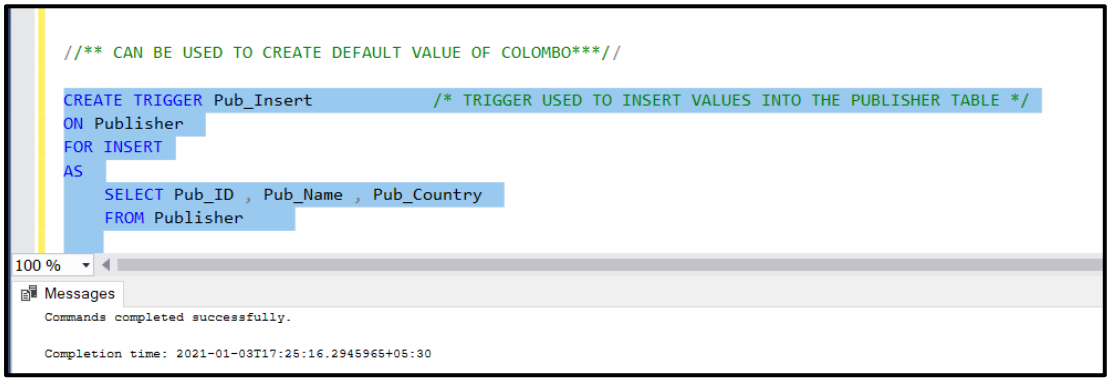
```
CREATE TRIGGER deletestaff
ON StaffTable
AFTER DELETE
AS
BEGIN
SELECT * FROM StaffTable
END

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	rohithagune32@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**



```
//** 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
```

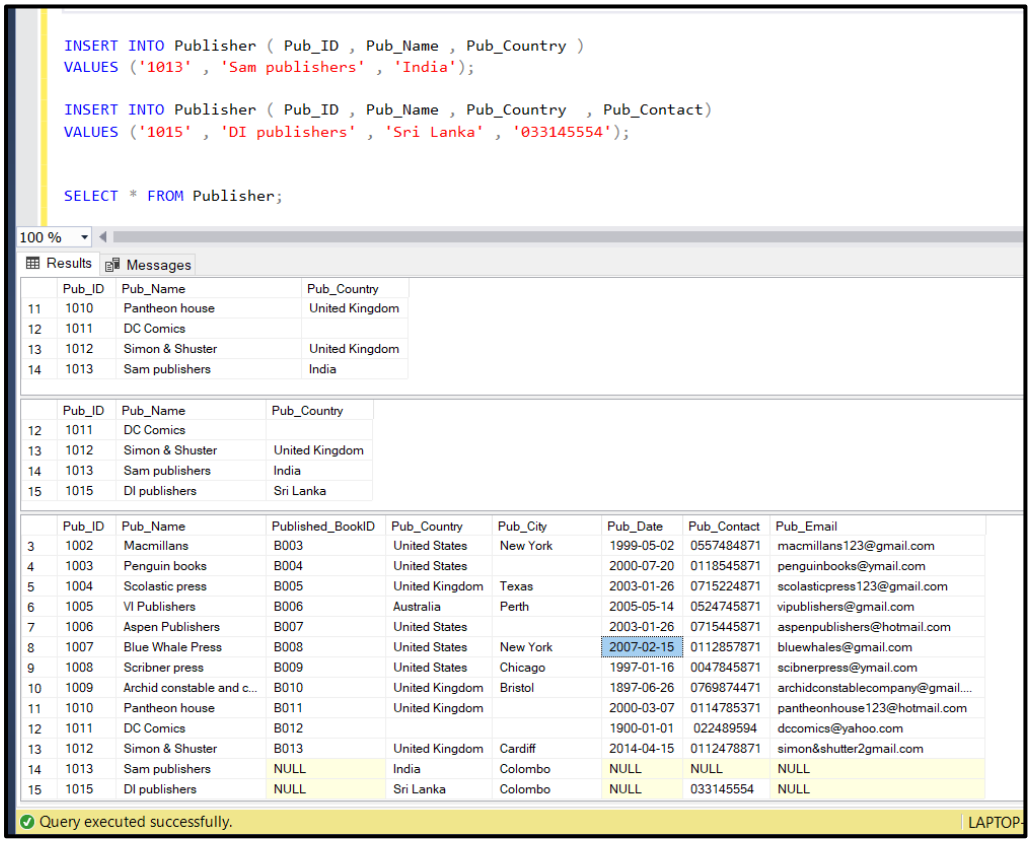
100 %

Messages

```
Commands completed successfully.

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 DFAULT Constraint of Column Pub_City.



```
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;
```

100 %

Results | Messages

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

	Pub_ID	Pub_Name	Pub_Country
12	1011	DC Comics	
13	1012	Simon & Shuster	United Kingdom
14	1013	Sam publishers	India
15	1015	DI publishers	Sri Lanka

	Pub_ID	Pub_Name	Published_BookID	Pub_Country	Pub_City	Pub_Date	Pub_Contact	Pub_Email
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	scibnerpress@ymail.com
10	1009	Archid constable and c...	B010	United Kingdom	Bristol	1897-06-26	0769874471	archidconstablecompany@gmail....
11	1010	Pantheon house	B011	United Kingdom		2000-03-07	0114785371	pantheonhouse123@hotmail.com
12	1011	DC Comics	B012			1900-01-01	022489594	dccomics@yahoo.com
13	1012	Simon & Shuster	B013	United Kingdom	Cardiff	2014-04-15	0112478871	simon&shutter2gmail.com
14	1013	Sam publishers	NULL	India	Colombo	NULL	NULL	NULL
15	1015	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 %

Results | Messages

	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;
```

0 %

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 %

Results | Messages

	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	scibnerpress@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&shutter2gmail.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
```

46 %

Results | Messages

	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	leam 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 %

Results | Messages

	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;
```

00 %

Results | Messages

	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;
```

100 %

Results | Messages

	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

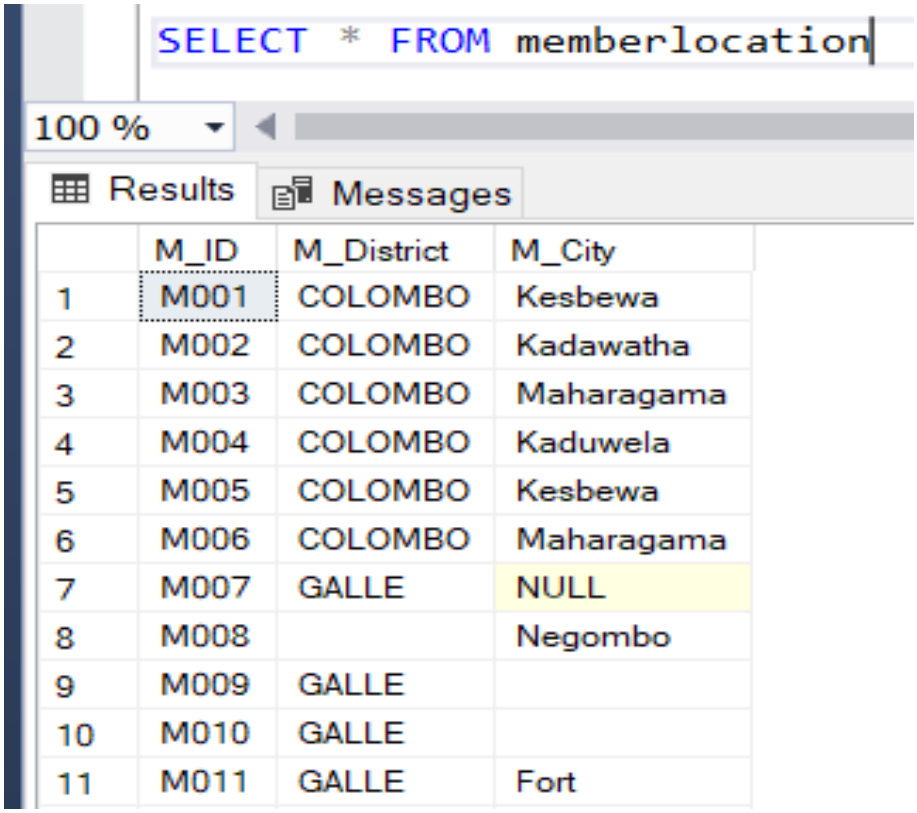
2) BORROWER TABLE

DATABASE VEIWS FOR BORROWER

VEIW 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

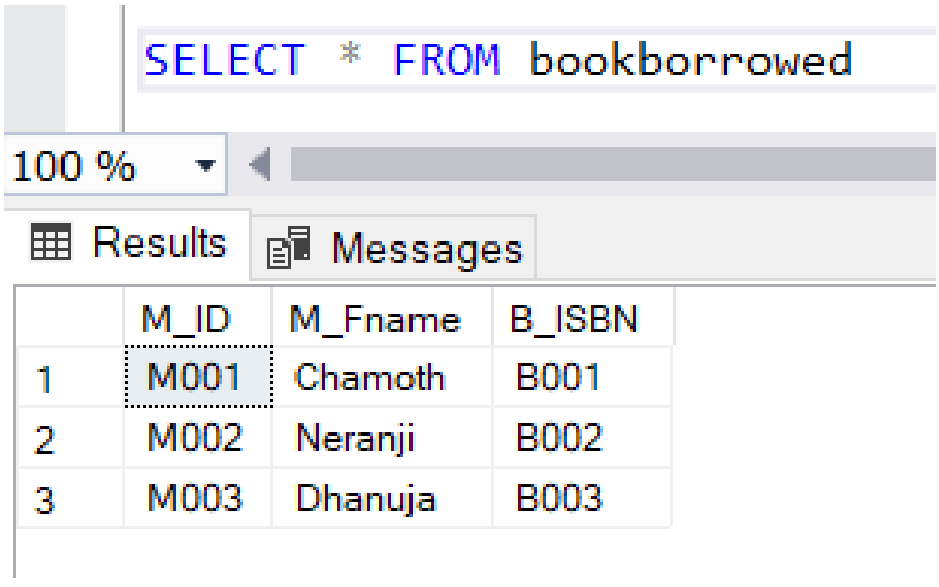


	M_ID	M_District	M_City
1	M001	COLOMBO	Kesbewa
2	M002	COLOMBO	Kadawatha
3	M003	COLOMBO	Maharagama
4	M004	COLOMBO	Kaduwela
5	M005	COLOMBO	Kesbewa
6	M006	COLOMBO	Maharagama
7	M007	GALLE	NULL
8	M008		Negombo
9	M009	GALLE	
10	M010	GALLE	
11	M011	GALLE	Fort

VEIW 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:

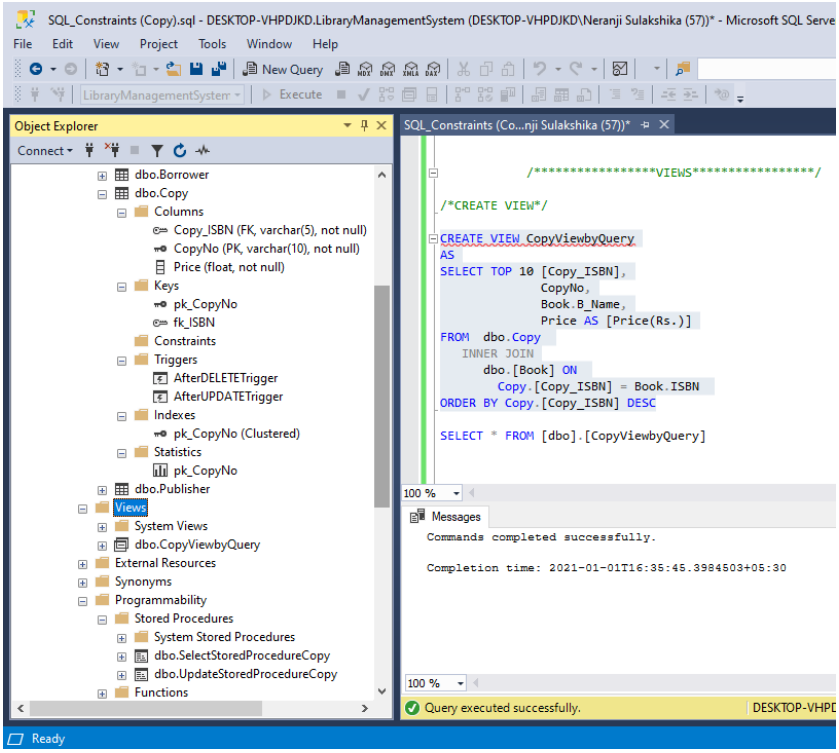


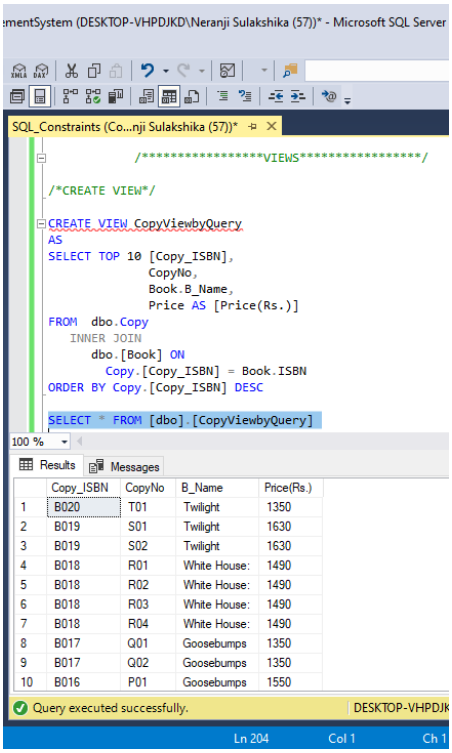
	M_ID	M_Fname	B_ISBN
1	M001	Chamoth	B001
2	M002	Neranji	B002
3	M003	Dhanuja	B003

3) COPY TABLE

1. Create View

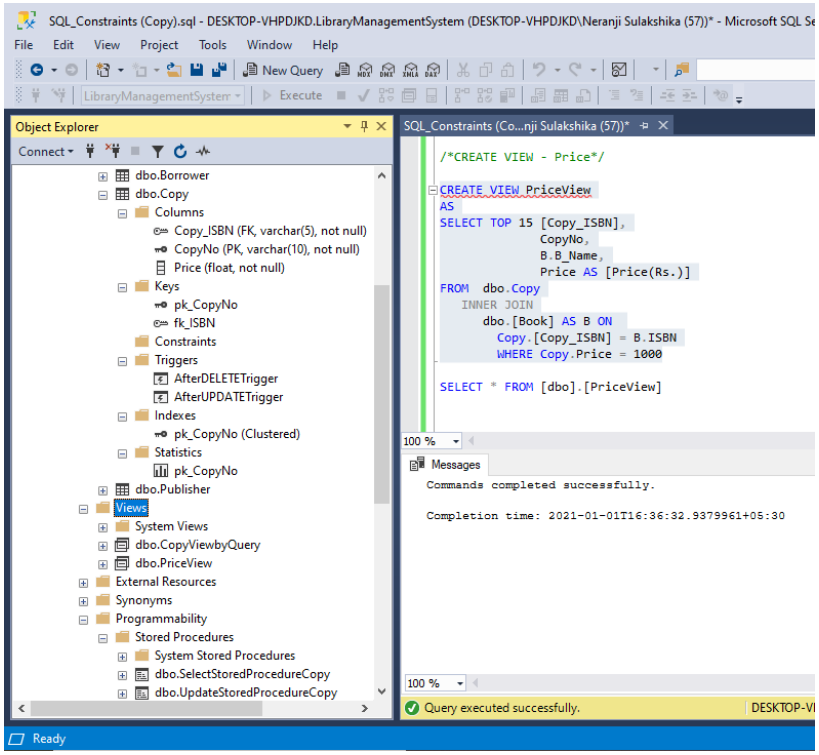
- Create and execute the ‘CopyViewQuery’ to view books copy information.

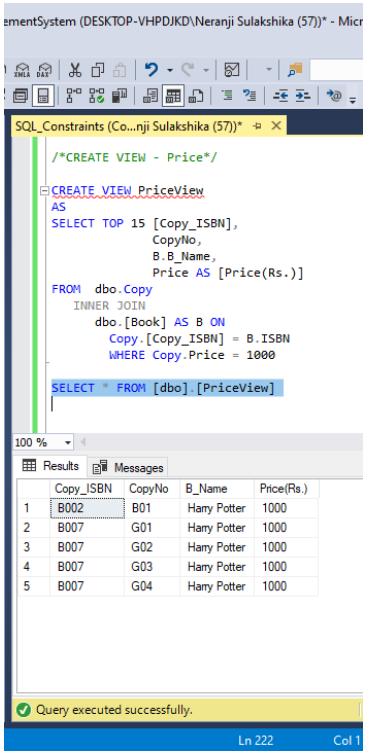




2. Create View

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

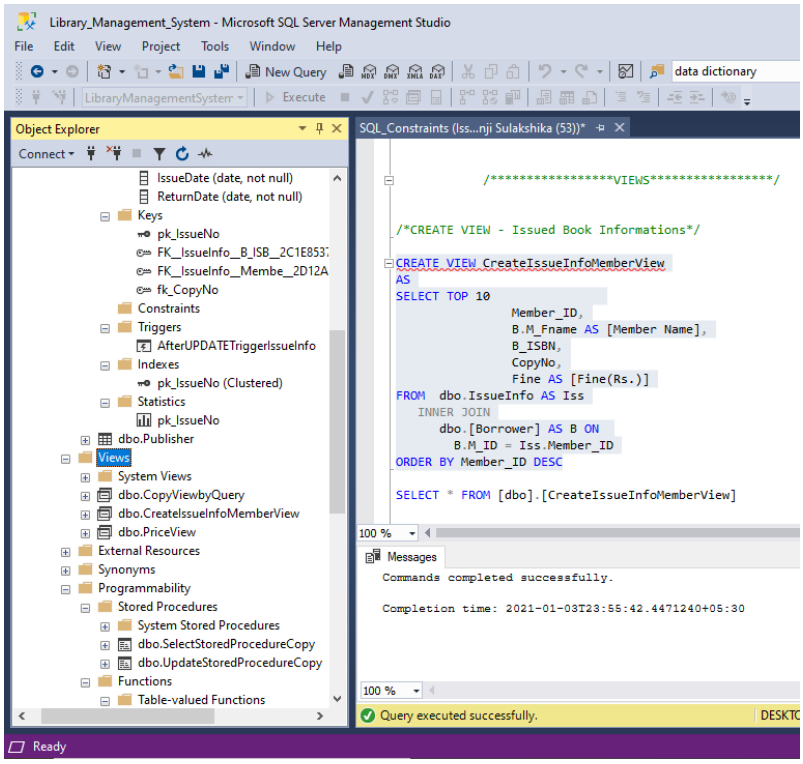


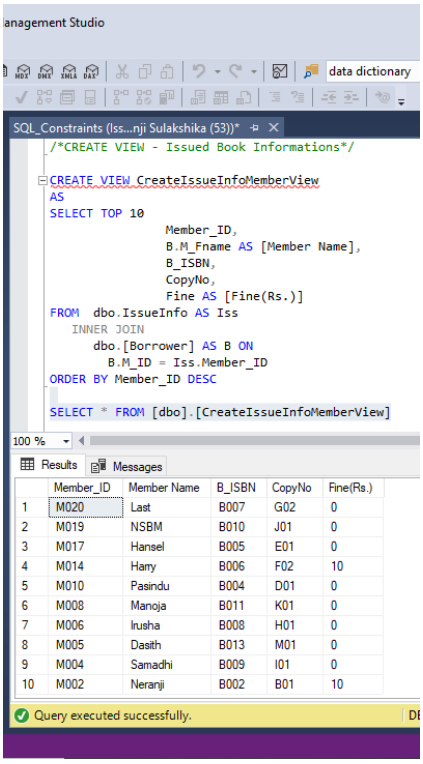


4) ISSUE-INFO TABLE

1. Create View

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

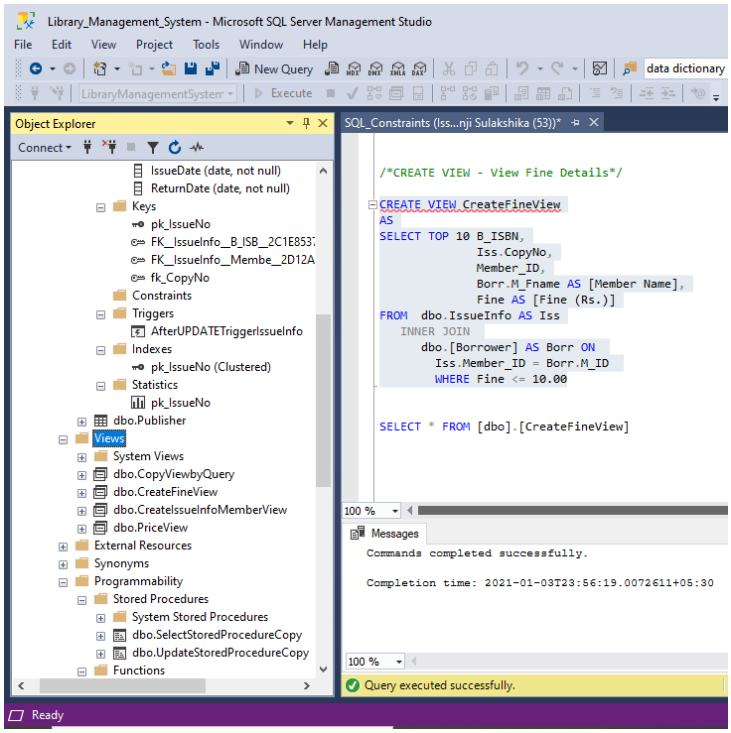


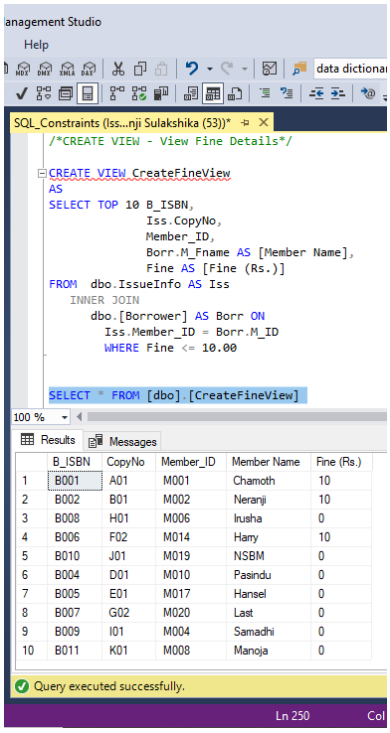


	Member_ID	Member Name	B_ISBN	CopyNo	Fine(Rs.)
1	M020	Last	B007	G02	0
2	M019	NSBM	B010	J01	0
3	M017	Hansel	B005	E01	0
4	M014	Harry	B006	F02	10
5	M010	Pasindu	B004	D01	0
6	M008	Manoja	B011	K01	0
7	M006	Irusha	B008	H01	0
8	M005	Dasith	B013	M01	0
9	M004	Samadhi	B009	I01	0
10	M002	Neranji	B002	B01	10

2. Create View

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





	B_ISBN	CopyNo	Member_ID	Member Name	Fine (Rs.)
1	B001	A01	M001	Chamoth	10
2	B002	B01	M002	Neranji	10
3	B008	H01	M006	Irusha	0
4	B006	F02	M014	Harry	10
5	B010	J01	M019	NSBM	0
6	B004	D01	M010	Pasindu	0
7	B005	E01	M017	Hansel	0
8	B007	G02	M020	Last	0
9	B009	I01	M004	Samadhi	0
10	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.

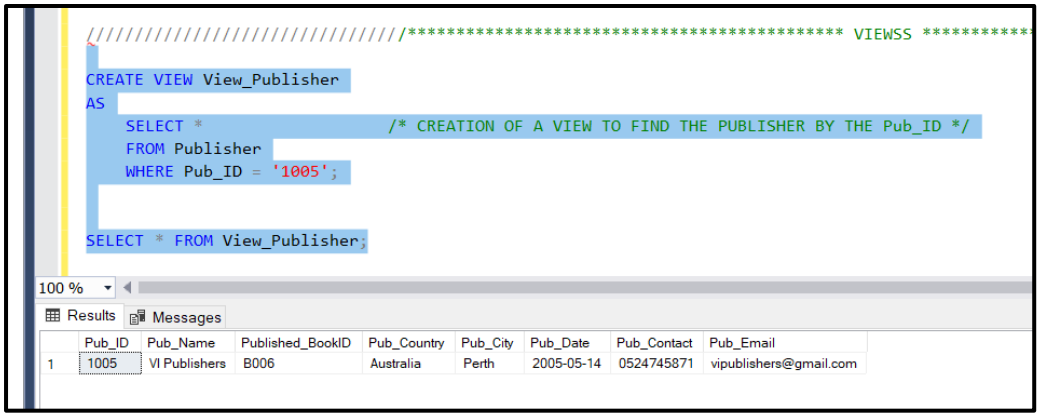
Results Messages

	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.



```
////////////////////////////////////////////// VIEWSS ***********

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;
```

100 %

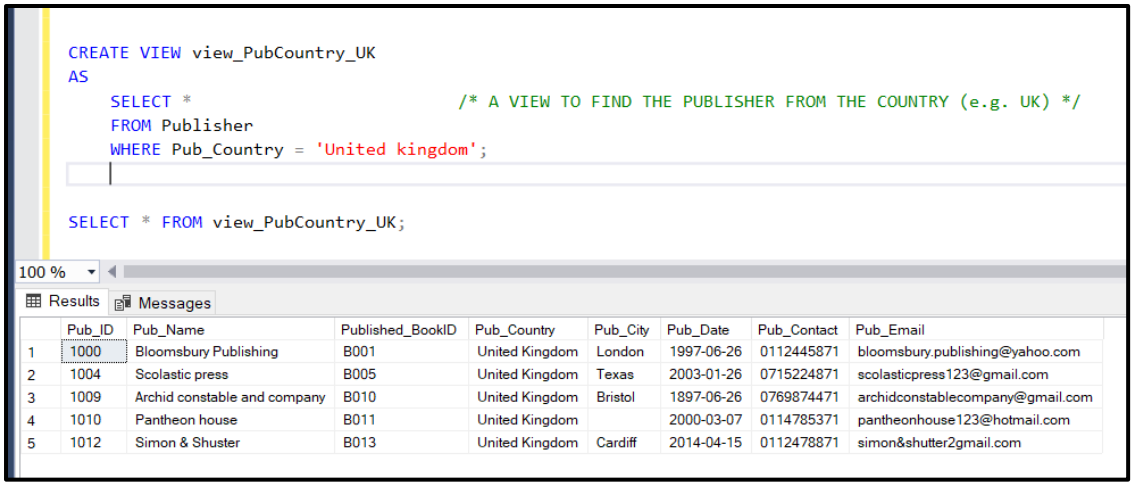
Results | Messages

	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



```
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;
```

100 %

Results | Messages

	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&shutter2gmail.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

```
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;
```

100 %

Results | Messages

	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');
```

100 %

Results | Messages

	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');
```

00 %

Results | Messages

	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.

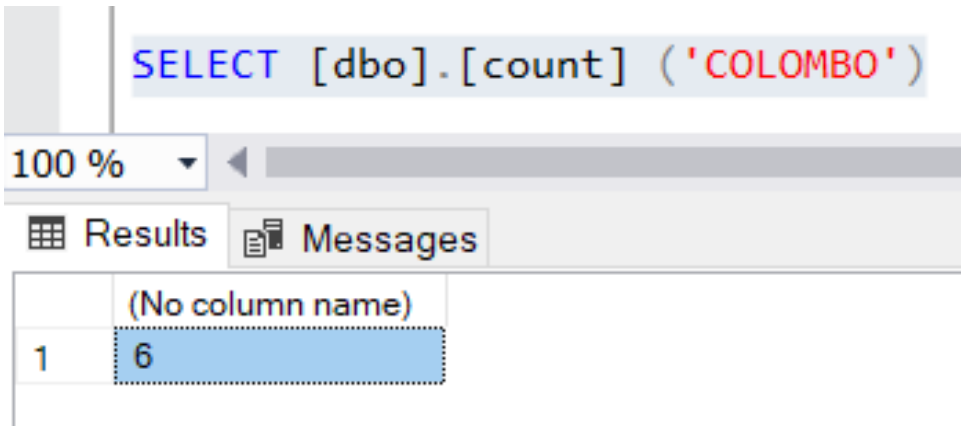
2) BORROWER TABLE

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')
```

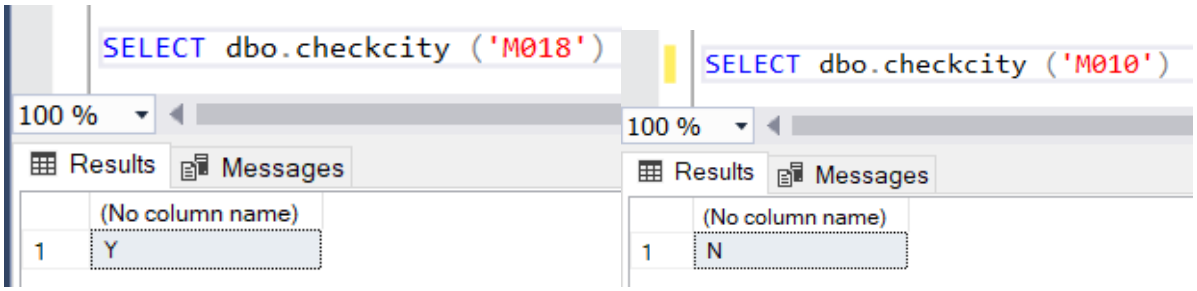
- By using this function, we can find out the count of members who comes from a particular district a value needs to be passed either its COLOMBO, GALLE OR JAFFNA. Then the count will be displayed accordingly.
- Execution are as follows



FUNCTION TO CHECK WETHER 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')
```

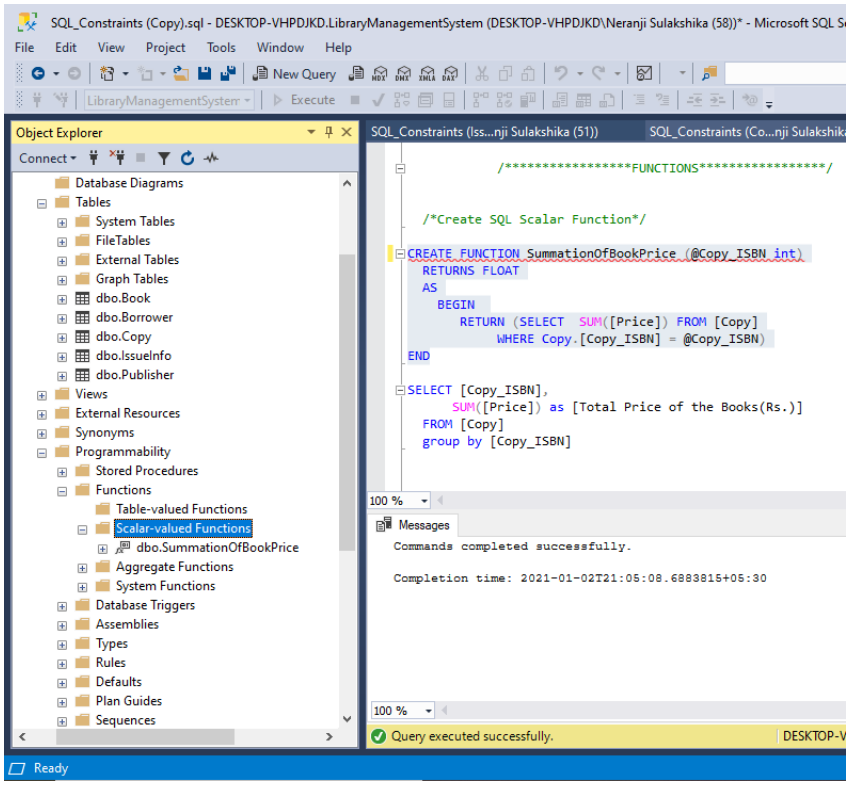
- By using this function we can find out the wether a given members city is given or NULL this can be simply done by passing the members Membership ID into the statement and if its Null its YES and if its NOT NULL then NO.
- Execution are as follows.

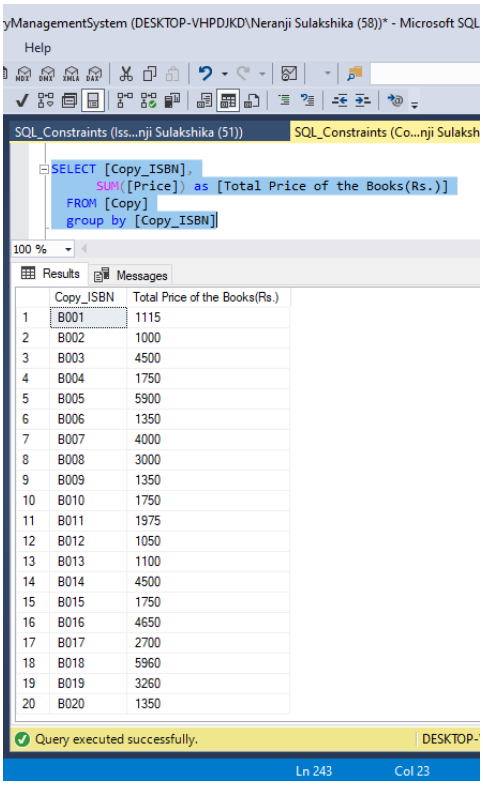


3) COPY TABLE

1. Create SQL Scalar Functions.

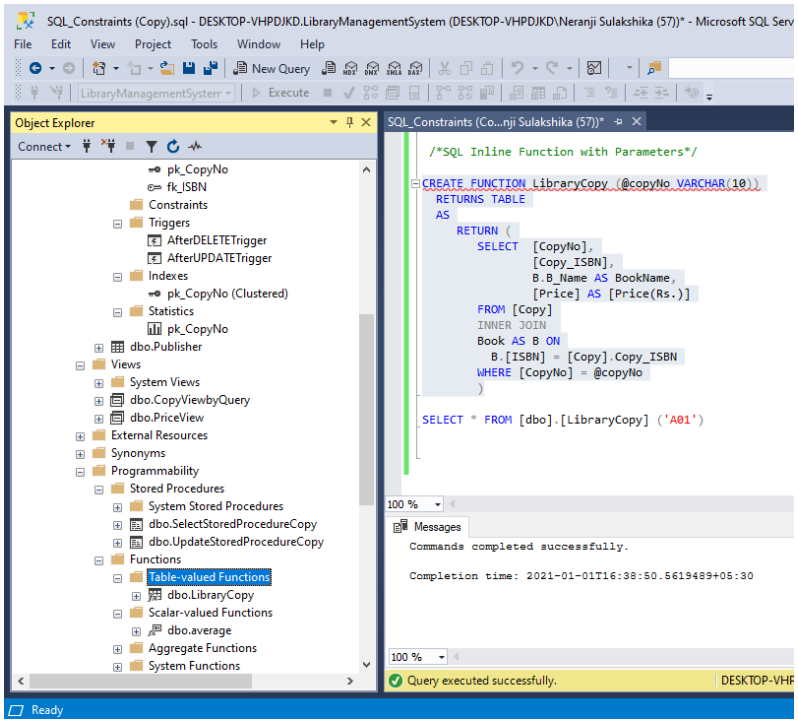
- Create and execute the ‘SummationOfBookPrice' function, to take Summation of book prices.

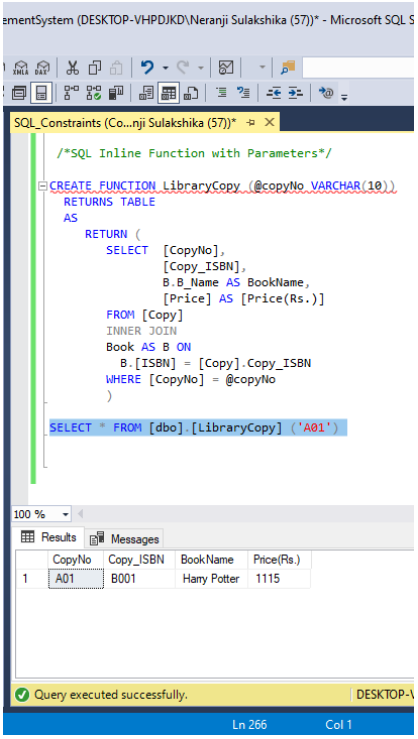




2. Create SQL Scalar Functions.

- Create and execute the ‘LibraryCopy’ function.

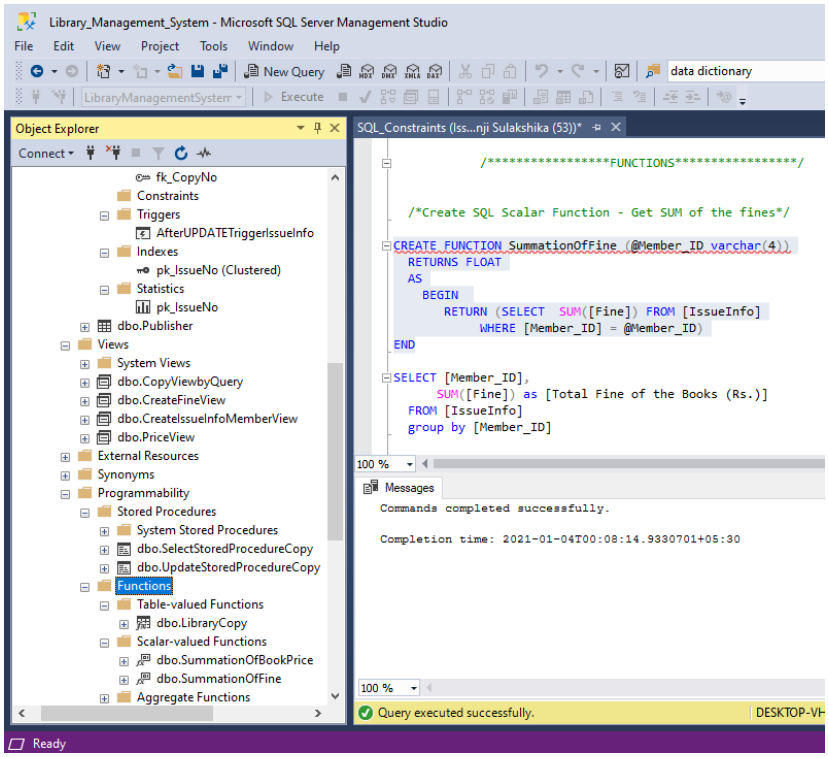




4) ISSUE-INFO TABLE

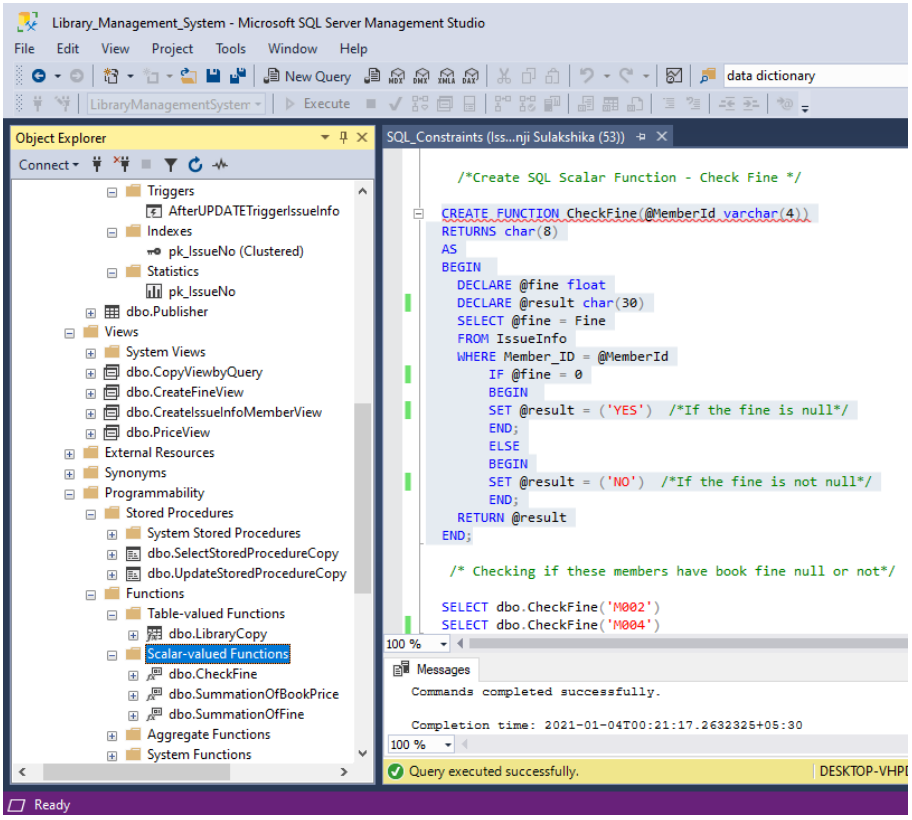
1. Create SQL Scalar Functions.

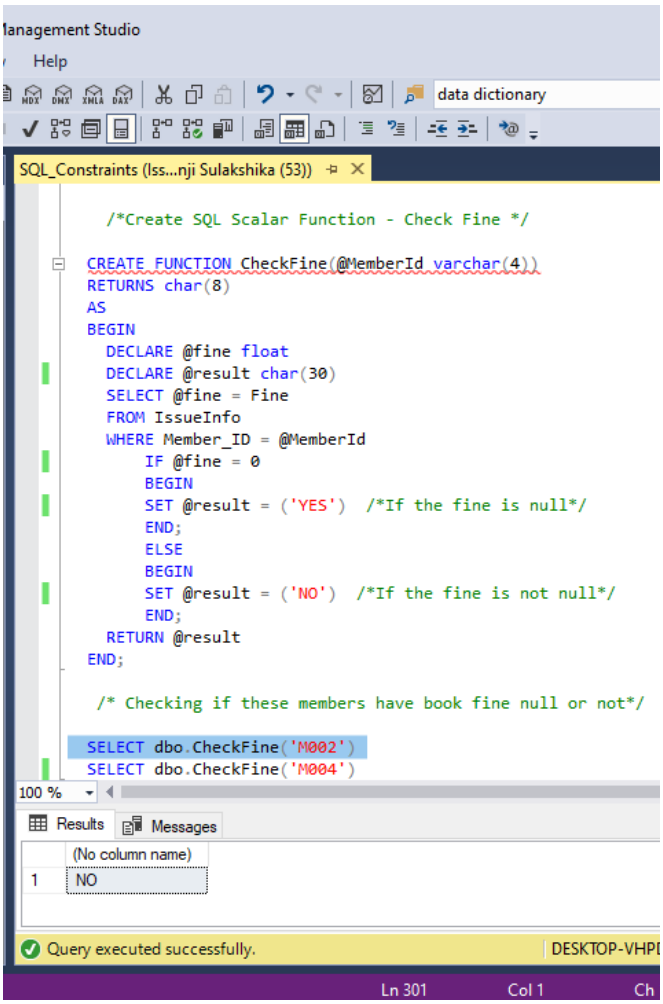
- Create and execute the ‘SummationOfFine' function, to take summation of the fine.

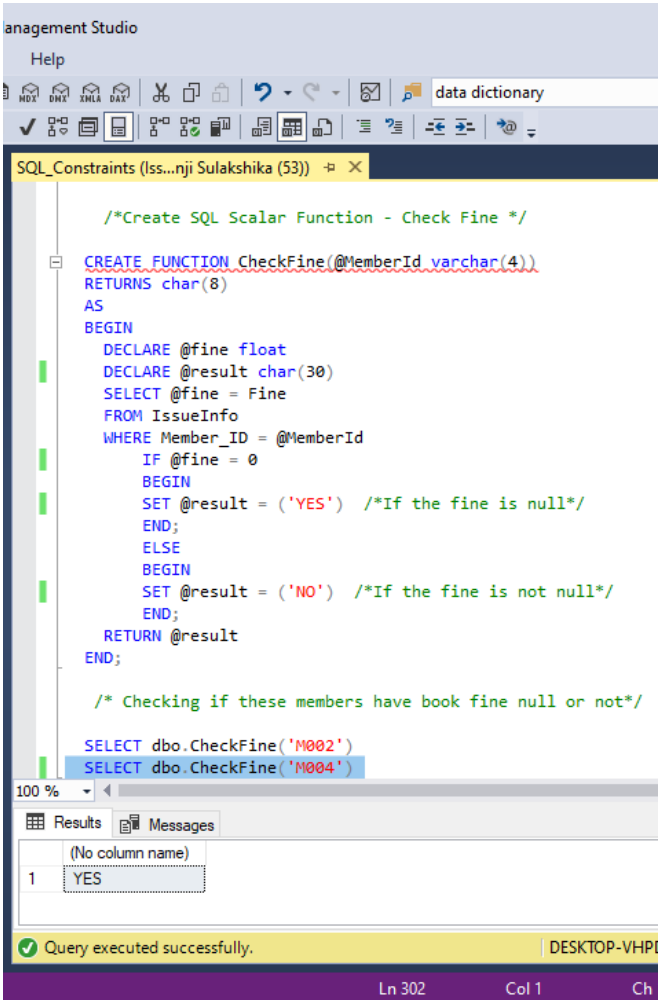


2. Create SQL Scalar Functions.

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




Help
data dictionary
SQL_Constraints (Iss...nji Sulakshika (53))
/*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.
Ln 301
Col 1


Help
data dictionary
SQL_Constraints (Iss...nji Sulakshika (53))
/*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.
Ln 302
Col 1

5) STAFF TABLE

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

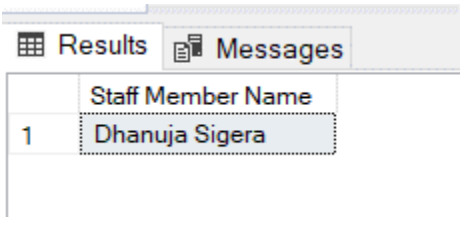
Creating functions can make yourself easy to search information just hitting by a desired key like an example as Staff ID,

```
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,



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

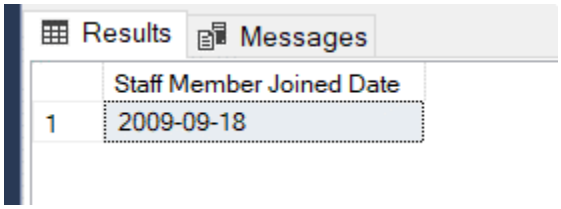
Finally, I created another function to perform the execution of joined date of a staff members just entered within staff id,

```
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,

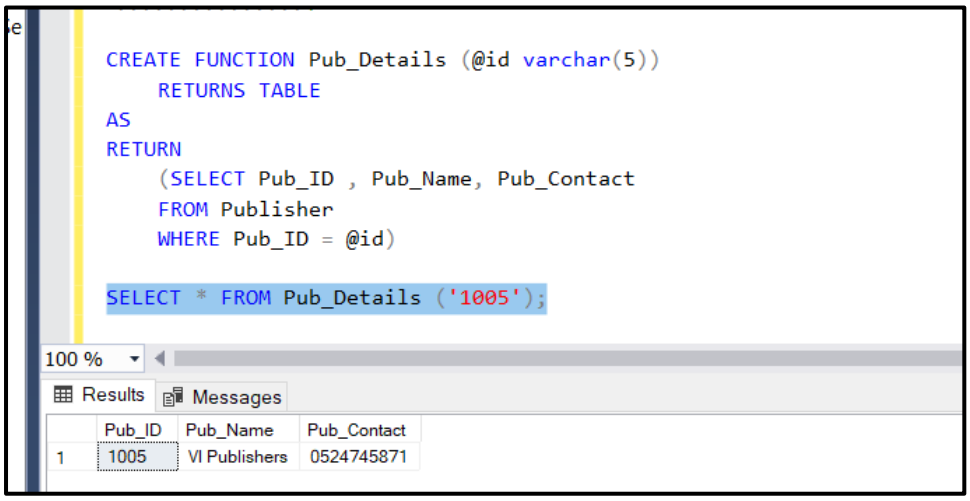
```
SELECT dbo.findjoineddate(7) AS [Staff Member Joined Date]
```

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



6) PUBLISHER TABLE

- **Creating a function**



```
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');
```

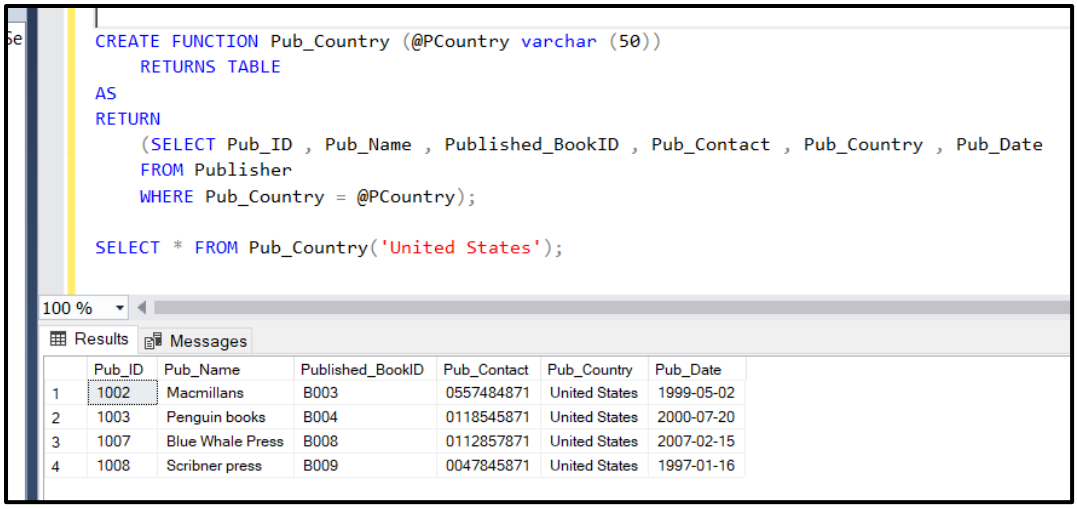
100 %

Results | Messages

	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**



```
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');
```

100 %

Results | Messages

	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

```
/***************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';
```

100 %

Results | Messages

	ISBN	B_Name	B_Title
1	B016	Goosebumps	Welcome to Dead House

	ISBN	B_Name	B_Title
1	B017	Goosebumps	Night of the Living Dummy

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

```
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
```

100 %

Messages

```
20

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

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

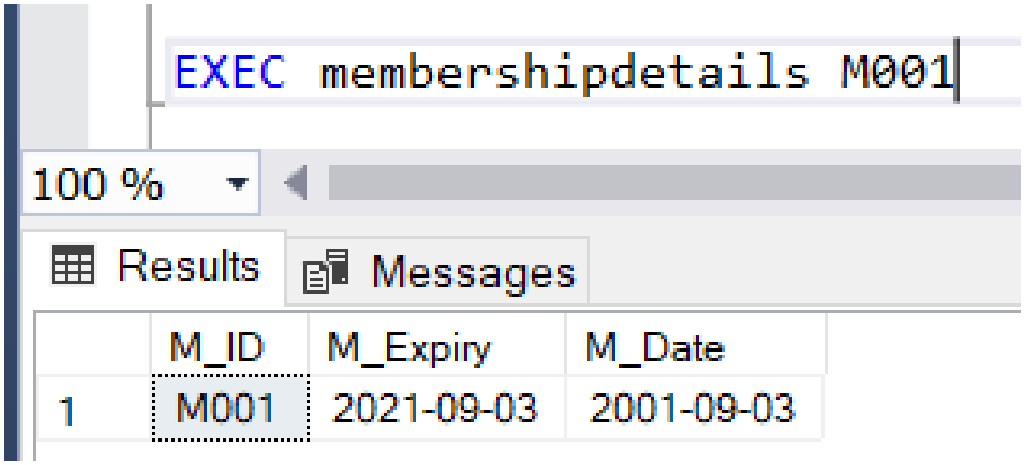
2) BORROWER TABLE

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
```

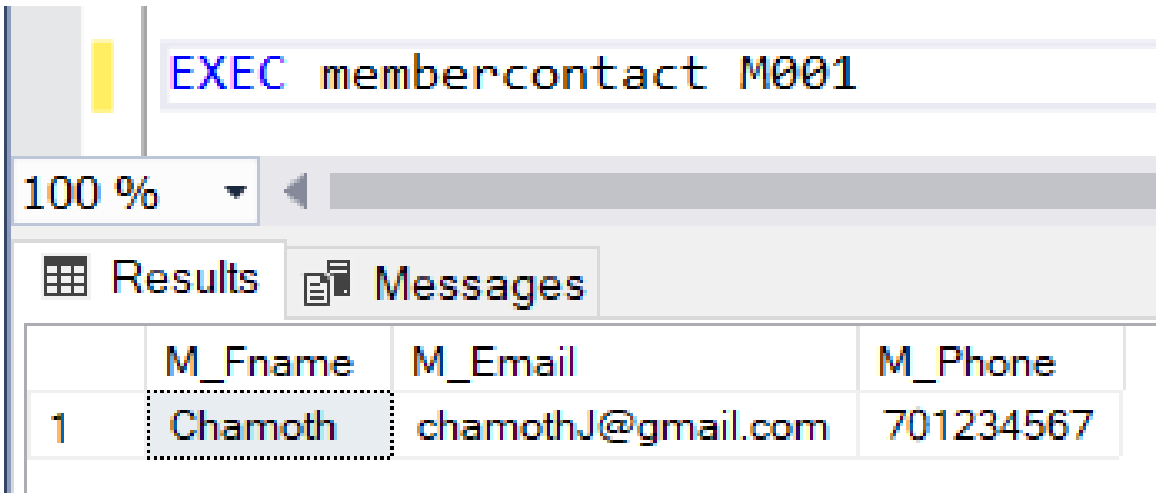
- We can use this procedure to input a members ID and get his/her membership details as it showcase the members joined date and membership expiry date.
- Execution details are as follows:



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
```

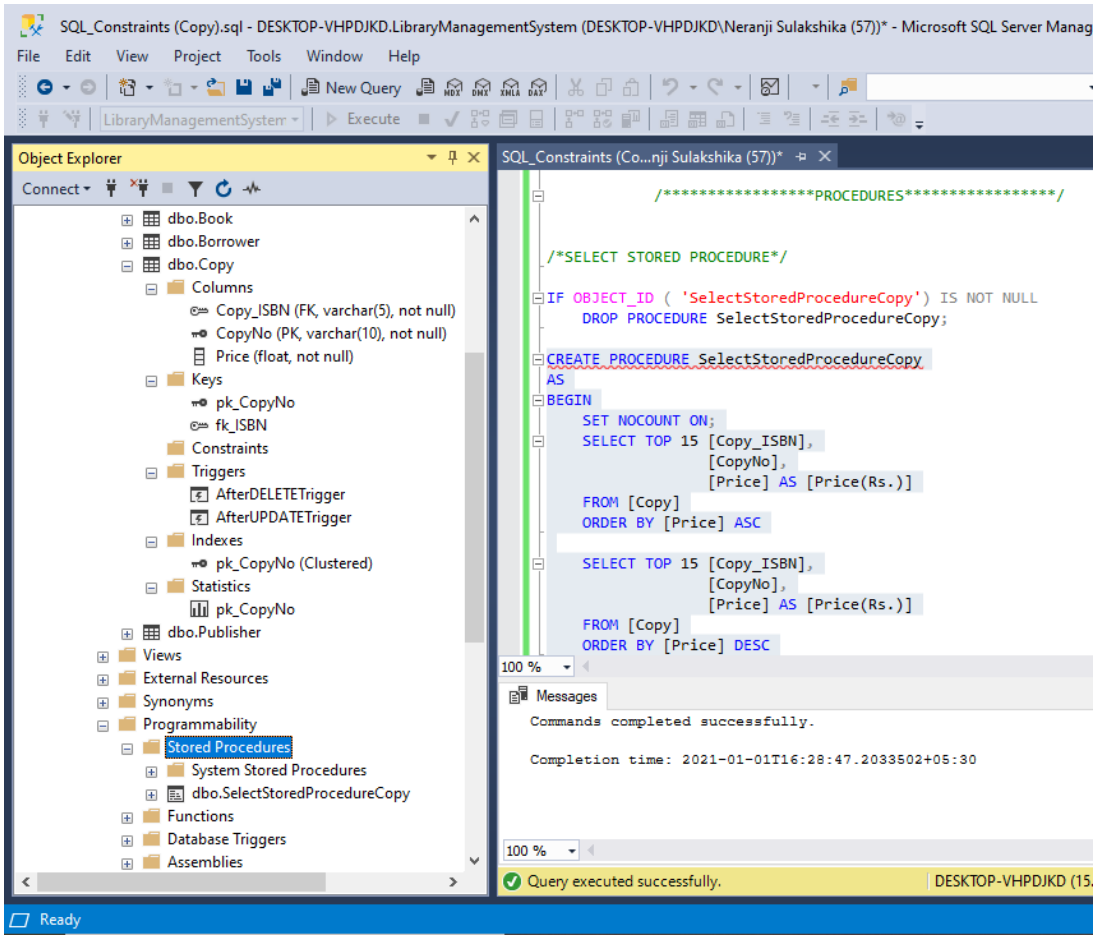
- We can use this procedure to find out a particular member contact information so when a membership ID is being passed the related member will be displayed along with the email address and telephone number of the borrower.



3) COPY TABLE

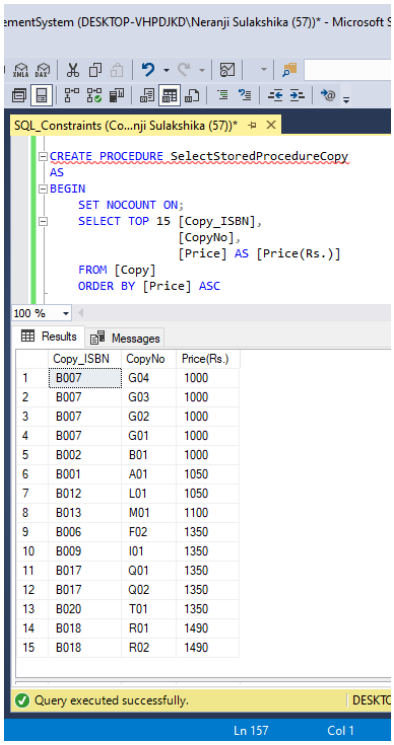
1. Select Stored Procedure.

- Create and execute the ‘SelectStoredProcedure’.

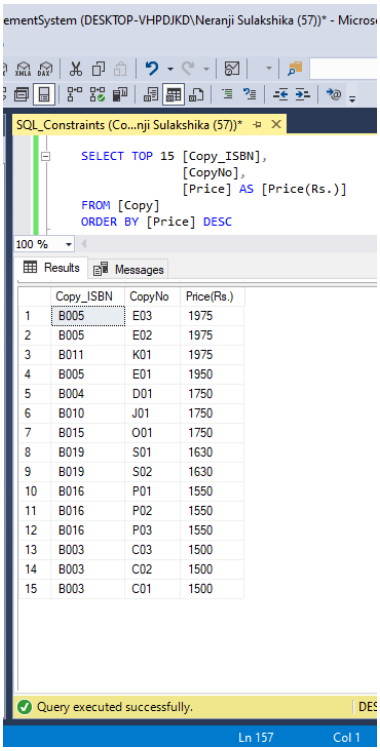


- Creating and executing the select stored procedure.

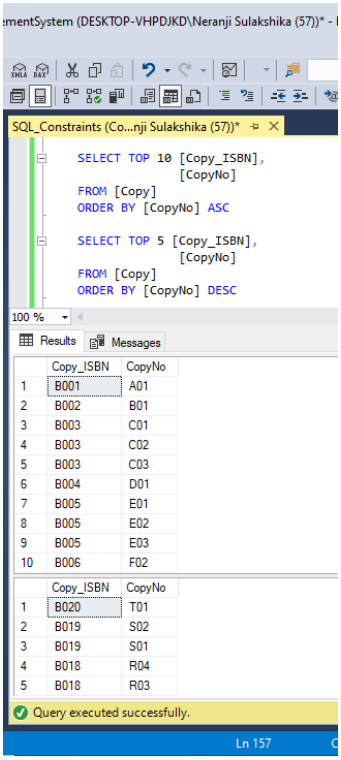
I. Execute the select stored procedure order by Price with ASC order.



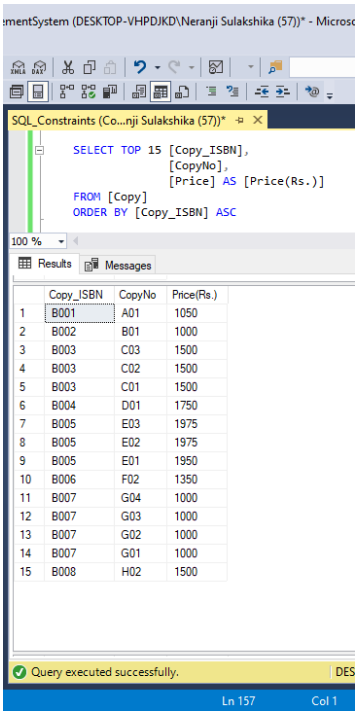
II. Execute the select stored procedure order by Price with DESC order.



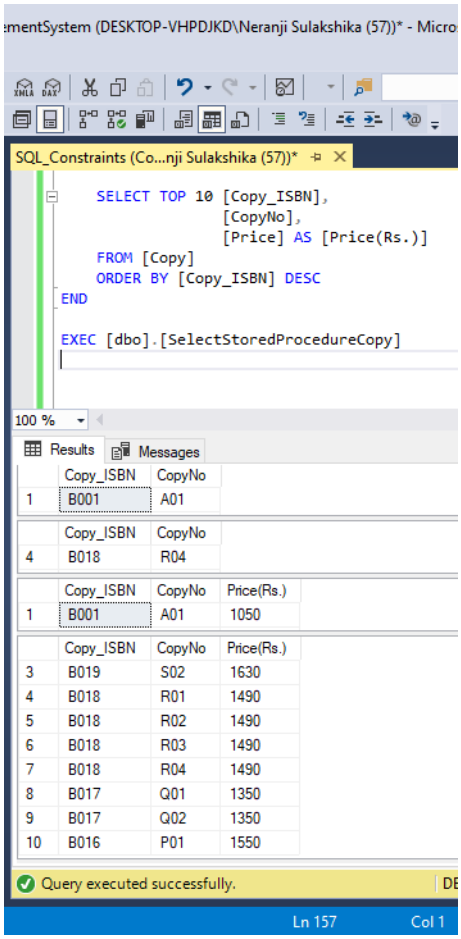
III. Execute the select stored procedure procedure order by CopyNo with ASC & DESC order.



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

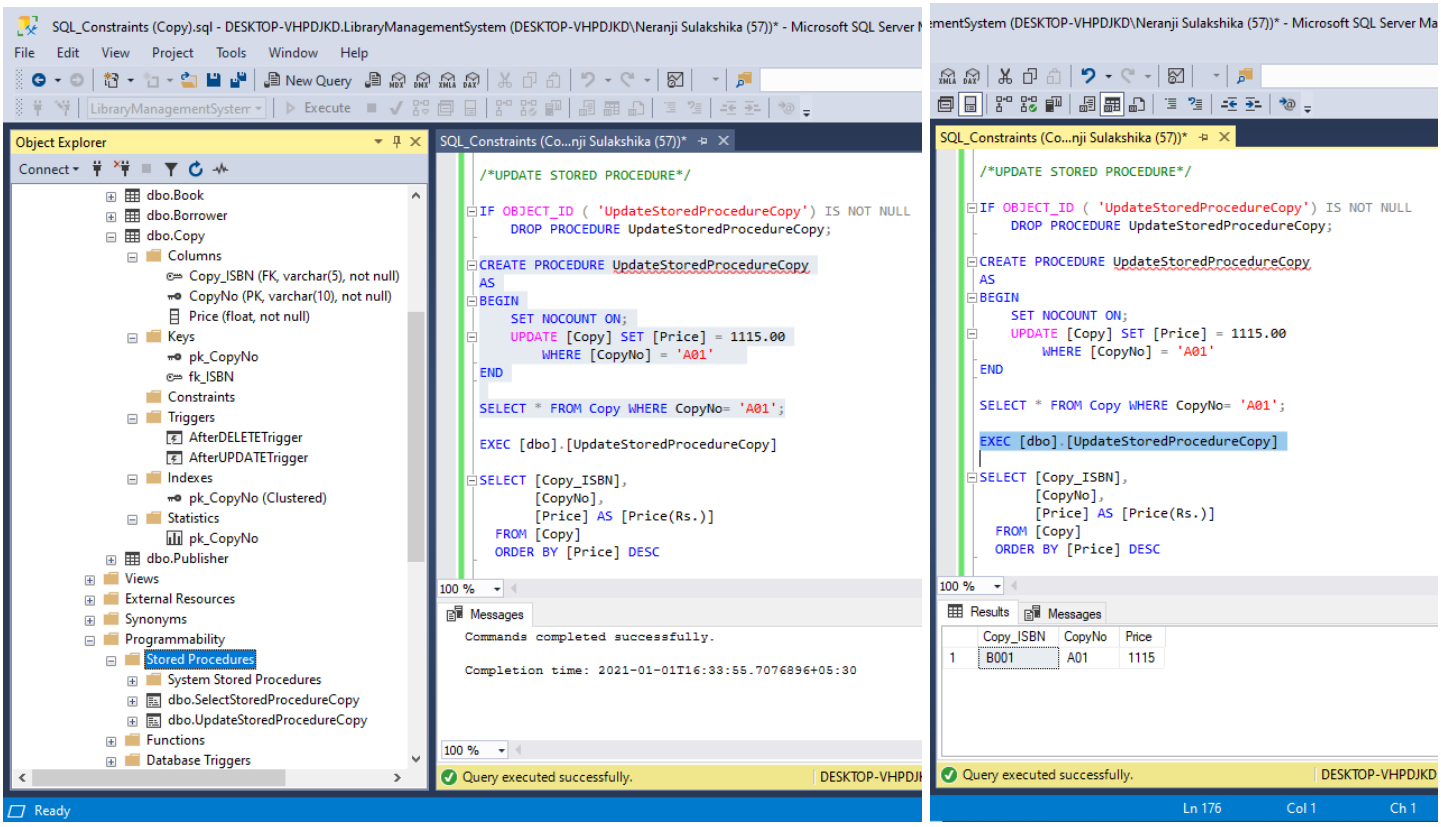


V. Execute the select stored procedure procedure order by Price with ASC order.

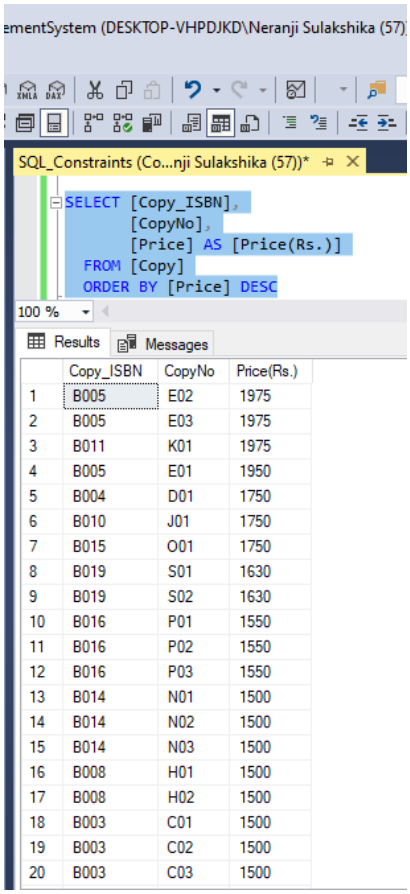


2. Update Stored Procedure.

- Creating and executing the update stored procedure.



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

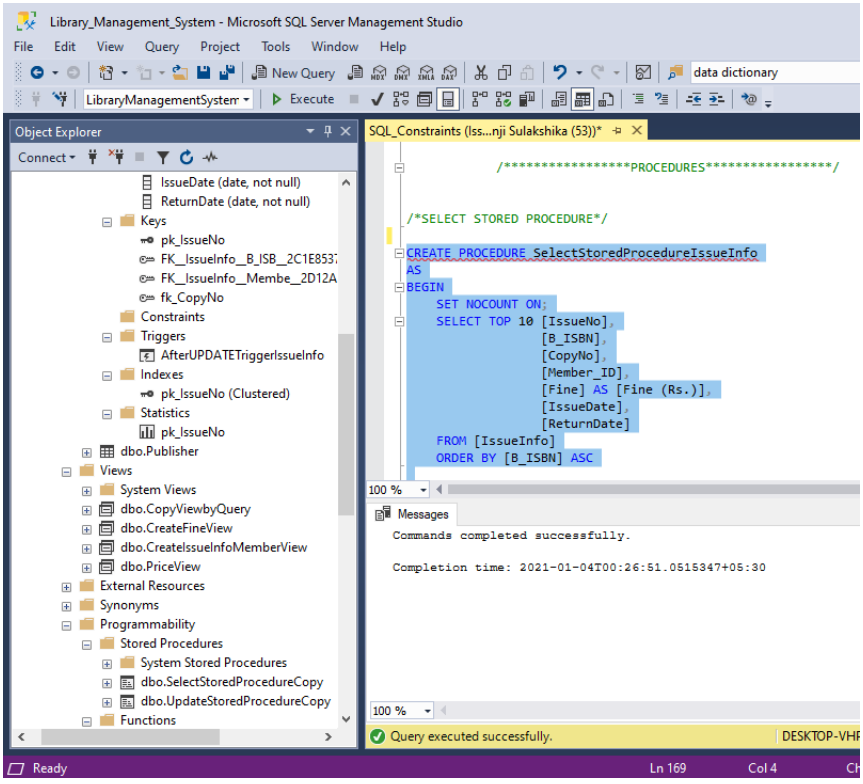


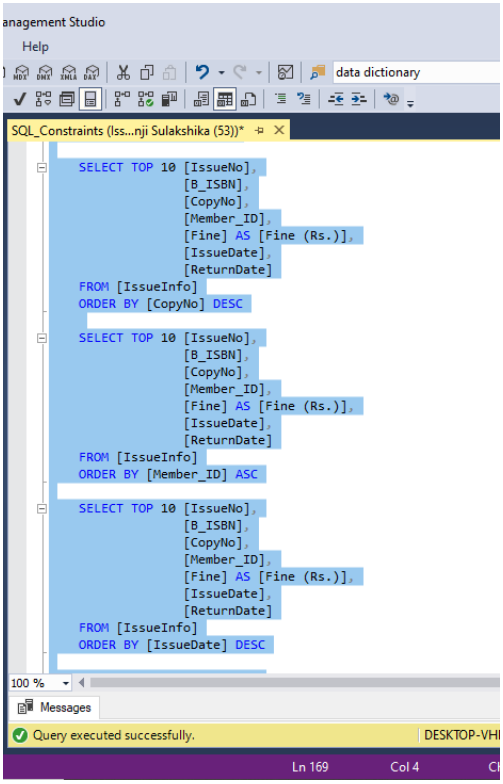
	Copy_ISBN	CopyNo	Price(Rs.)
1	B005	E02	1975
2	B005	E03	1975
3	B011	K01	1975
4	B005	E01	1950
5	B004	D01	1750
6	B010	J01	1750
7	B015	O01	1750
8	B019	S01	1630
9	B019	S02	1630
10	B016	P01	1550
11	B016	P02	1550
12	B016	P03	1550
13	B014	N01	1500
14	B014	N02	1500
15	B014	N03	1500
16	B008	H01	1500
17	B008	H02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

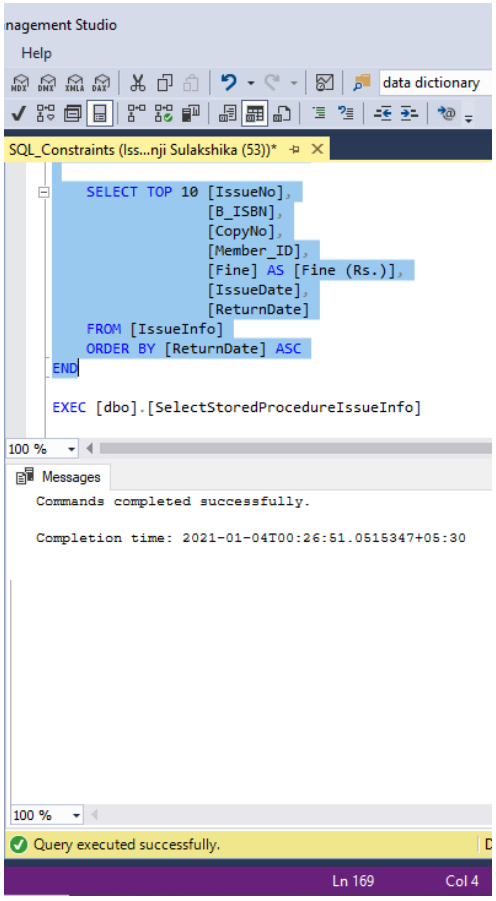
Query executed successfully.
Ln 181

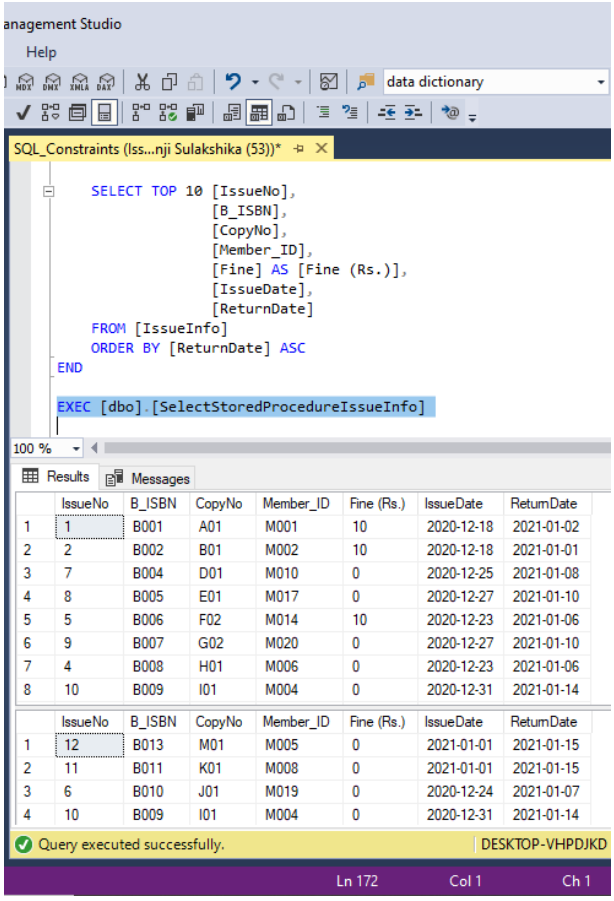
4) ISSUE-INFO TABLE

1. Select Stored Procedure.



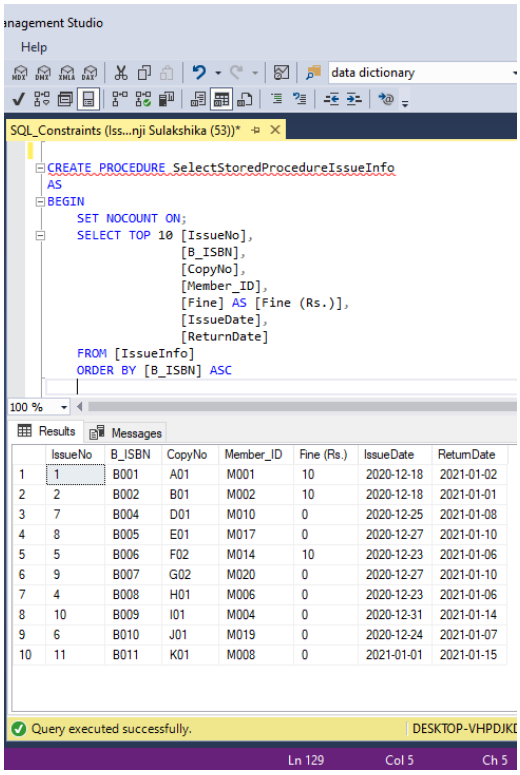






➢ Creating and executing the select stored procedure.

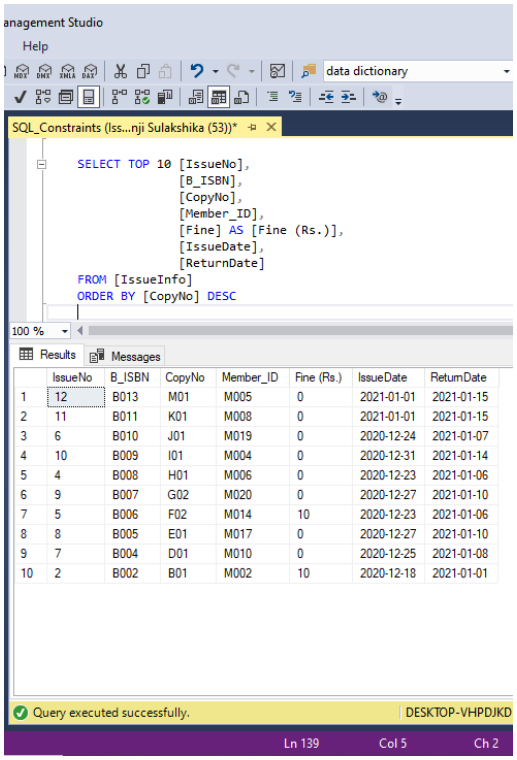
I. Execute the select stored procedure order by B_ISBN with ASC order.



```
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
```

	IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	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	7	B004	D01	M010	0	2020-12-25	2021-01-08
4	8	B005	E01	M017	0	2020-12-27	2021-01-10
5	5	B006	F02	M014	10	2020-12-23	2021-01-06
6	9	B007	G02	M020	0	2020-12-27	2021-01-10
7	4	B008	H01	M006	0	2020-12-23	2021-01-06
8	10	B009	I01	M004	0	2020-12-31	2021-01-14
9	6	B010	J01	M019	0	2020-12-24	2021-01-07
10	11	B011	K01	M008	0	2021-01-01	2021-01-15

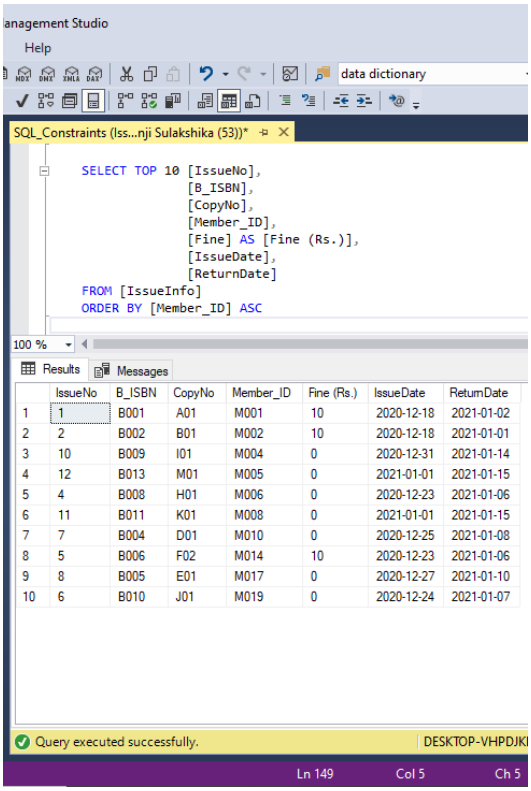
II. Execute the select stored procedure order by CopyNo with DESC order.



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

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

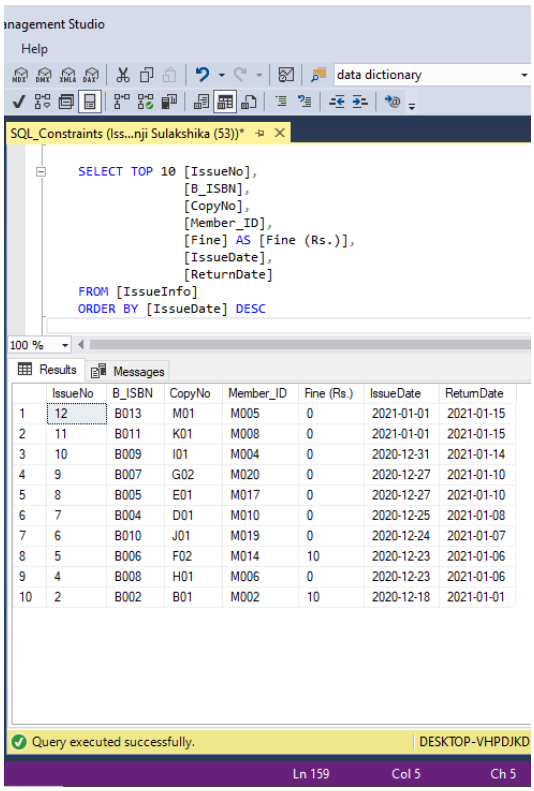
III. Execute the select stored procedure order by Member_ID with ASC order.



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

	IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	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	10	B009	I01	M004	0	2020-12-31	2021-01-14
4	12	B013	M01	M005	0	2021-01-01	2021-01-15
5	4	B008	H01	M006	0	2020-12-23	2021-01-06
6	11	B011	K01	M008	0	2021-01-01	2021-01-15
7	7	B004	D01	M010	0	2020-12-25	2021-01-08
8	5	B006	F02	M014	10	2020-12-23	2021-01-06
9	8	B005	E01	M017	0	2020-12-27	2021-01-10
10	6	B010	J01	M019	0	2020-12-24	2021-01-07

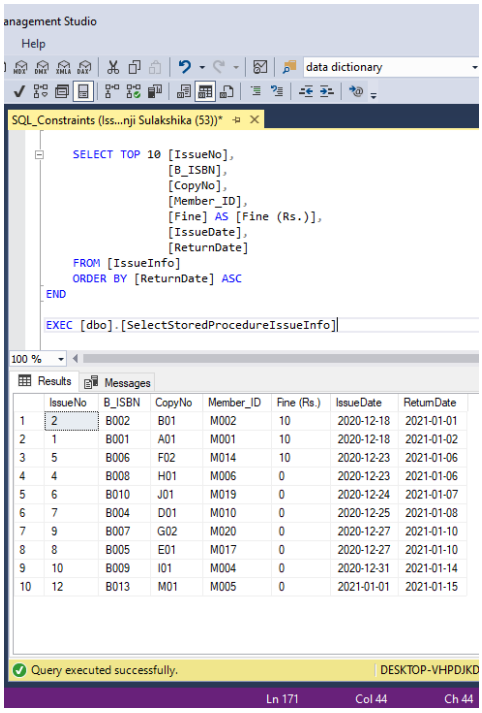
IV. Execute the select stored procedure order by IssueDate with DESC order.



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

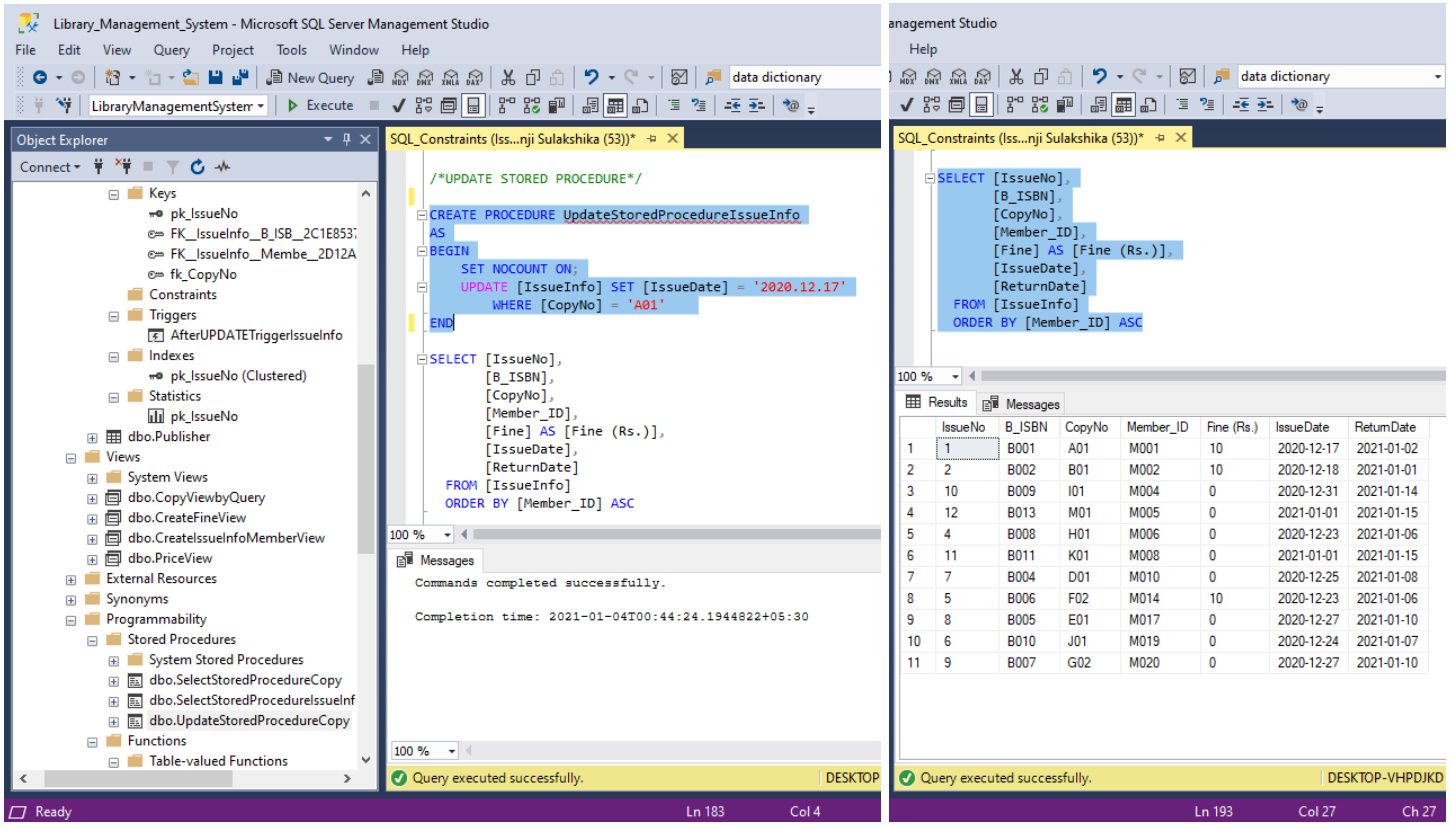
	IssueNo	B_ISBN	CopyNo	Member_ID	Fine (Rs.)	IssueDate	ReturnDate
1	12	B013	M01	M005	0	2021-01-01	2021-01-15
2	11	B011	K01	M008	0	2021-01-01	2021-01-15
3	10	B009	I01	M004	0	2020-12-31	2021-01-14
4	9	B007	G02	M020	0	2020-12-27	2021-01-10
5	8	B005	E01	M017	0	2020-12-27	2021-01-10
6	7	B004	D01	M010	0	2020-12-25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

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



2. Update Stored Procedure.

- Create and execute the update stored procedure.



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,

100 %

Results | Messages

	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
```

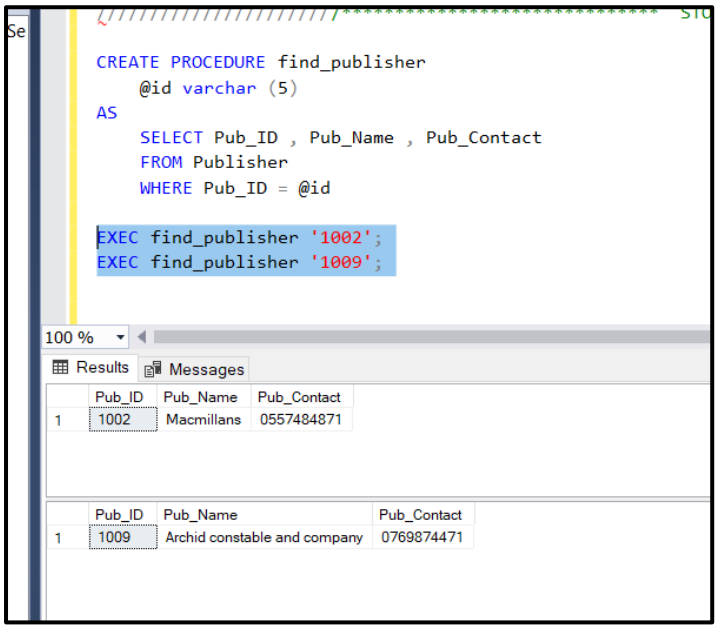
The result can be like this,

Results | Messages

	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

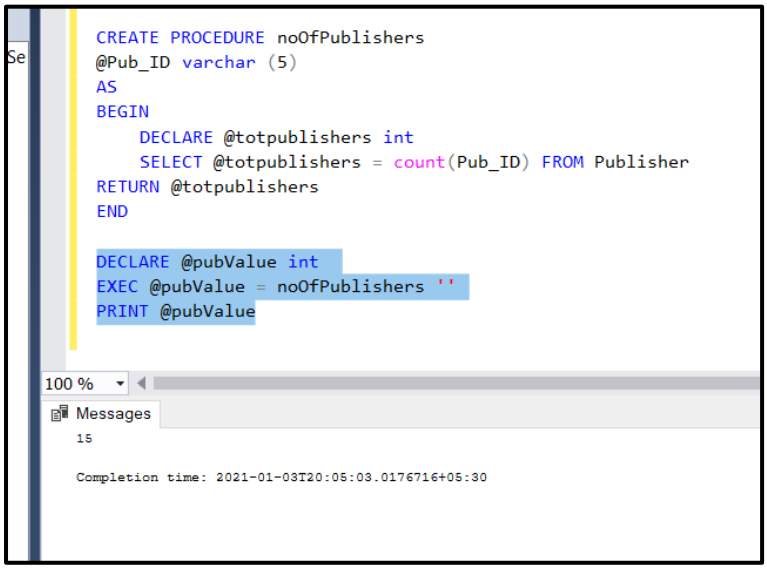
6) PUBLISHER TABLE

1. Procedure to find the publisher by ID



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 of publisher name and the publisher contact.

2. Function to find the number of publishers



- 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 EVALUVATION

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 involves 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.