IFT 530: Advanced Database Management Systems

LIBRARY MANAGEMENT SYSTEM

Aishwarya Ramaiah Kumar

Rishabh Singh

Kartavi Manishkumar Shah

Prof. Robert Rucker

December 4, 2022

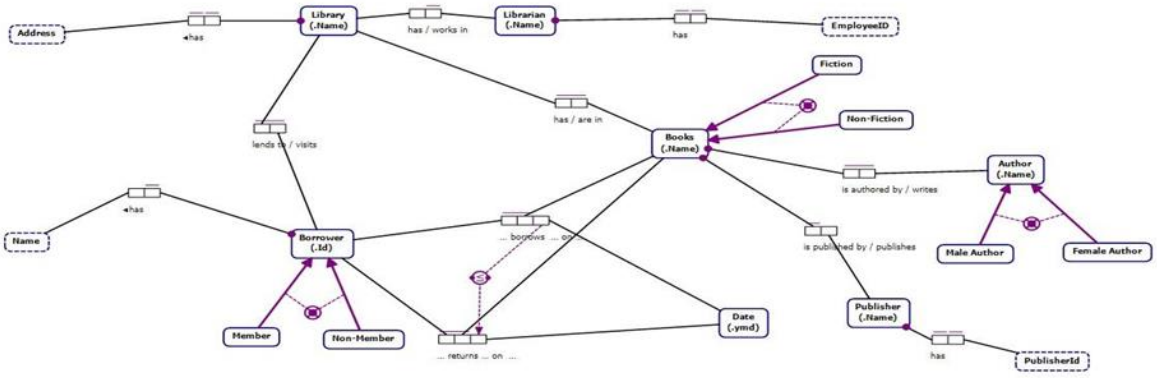
Section 0 -Summary

The Library Management Systems was the topic we settled on for our project. By simplifying the process of recording book records and getting data on books available, book transactions, and book holders, this project aids in the operation of libraries. A consumer can use this system to find out how many/what books are available in the library. Customers with memberships will be given a special ID that will be used to recognize them.

Library Management Systems has the following major functions:

- To track and determine the borrow and return dates for books, add or remove books from the book inventory, and more
- Keeping a record of each book's inventory, which will include details about each one and be organized by genre, publisher, and author.
- Creating user records that include information on both members and non-members as well as information on the employee

The model's entities are the library, the employee, the books, the clients, the authors, and the publishers.



Section 1 - Introduction

You can purchase books in paperback or digital form at a library. In order to keep track of who borrowed a book and when they were supposed to return it, libraries today are responsible for maintaining databases for all students, as well as for the teaching and non-teaching staff of the institution. Considering that all of these documents and databases were formerly handled manually.

Our project proposal is Library Management Systems. The main problem with manually retaining records was that they were unreliable and it took a lot of time and effort to trace a book.

Additionally, it was impossible to maintain track of every book that was issued and returned while still providing central access to them.

We are developing library administration systems that will only be accessible to authorized users, enabling a fully digital library service. As a result, it will be much simpler to enter book records and find out information about books, book transactions, and book holders. This method allows customers to check the number of books in the library and issue books to themselves. Customers and employees each have a unique perspective on this system. A special ID that enables identification will be supplied to customers who have purchased memberships.

The importance of BMS lies in how much simpler public and university bookstores will find it to keep track of patrons/students as well as books. Additionally, it will make it easier to track a book that has already been released. ASU has many libraries, thus this will help maintain a consolidated record that will be helpful for students in figuring out which library holds the needed book.

Library Management System has the following major functions:

- To change the number of books in the books inventory
- To keep track of and figure out when to return borrowed books
- Keeping track of each book's inventory, which will include details like its quantity, its description, and a classification based on the kind of book and its location
- A transactional record for the issued books is being marinated
- Establishing user records that include information on both members and guest users.

A model or blueprint is constructed to specify how data is organized, sorted, and modified in a database. The design takes into account a variety of elements, including where data will be stored, how it will be categorized, and how data from various database tables will interact.

Context

A library is a collection of various informational resources. These add-ons had created a distinct group of readers, students, and others who could consult or borrow the book more readily.

The library Database Management System is used to quickly locate books and gain access to journals. Traditional library processes are automated by the library automation system, which also lightens the workload of library staff.

It guarantees the consistency and accuracy of the data. People started to place a higher value on information, and technology transformed what information consumers expected from libraries. It assigns the librarian both a choice and a duty.

The integrated library system is used to perform more complicated tasks and enables staff to manage library resources more effectively, saving time and effort.

A library management system can help staff and patrons operate more effectively. Additionally, it makes it simpler for staff to organize books and keep track of those that have been borrowed, renewed, and not returned.

You may quickly reduce the workforce by employing a library management system while still storing various manual files electronically. A lot of data may be stored on one system, which reduces the need for manual files.

The following are the recognized entities for our project on a book store management system:

- **Employee:** Basic employer information includes the name and employee ID references in the employee entity. It has a one-to-many connection with the library since each employee can only work for one library, even if a library might have many employees.
- **Library:** Name and address of the store are references in the library entity.
- **Customer:**The necessary information must be submitted in order to issue a book, and the Member and Non-Member client entities contain references such name and id. There is a many-to-many link between this and libraries. A big number of libraries allow for the borrowing of items by a large number of patrons, and a large number of libraries allow for the visitation of a large number of patrons.
- **Books:** There are three names associated with books: Author, Book, and Publisher. Books are divided into two subtypes: fiction and non-fiction. This is a many-to-many relationship with the library. Many libraries have vast collections of books, and many libraries have large collections of books.
- **Publisher:** Properties like the publisher ID and name will be included in the publisher entity. It has a one-to-many link with books. While a single publisher can publish several books, they can only publish one book at a time.

- **Author:** The author object will have attributes like the author's name and separate fields for male and female authors. There is a many-to-many link between books and writers in that many authors can publish many books, and vice versa.

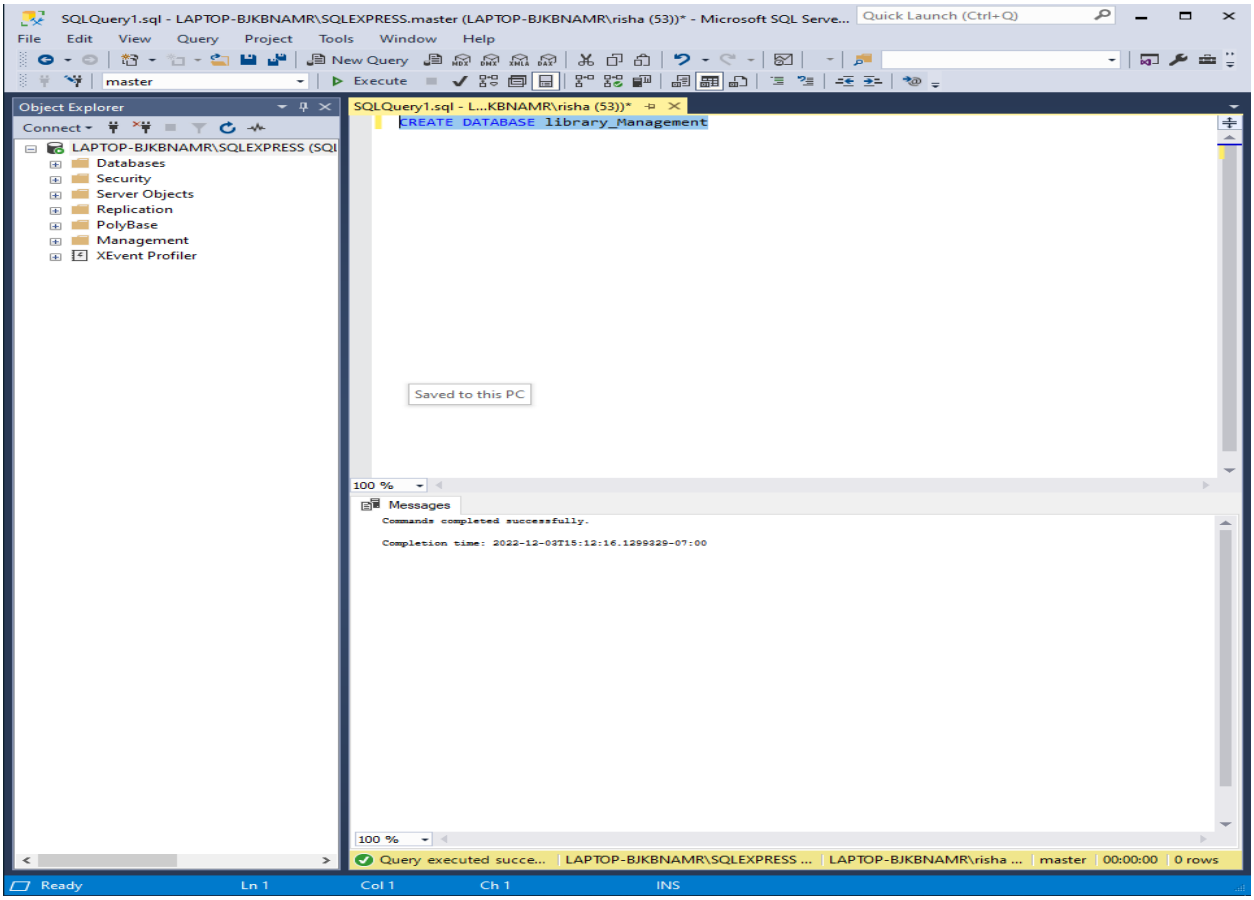
Queries which SQL database is answering:

- To create a library table:

  CODE:

  CREATE DATABASE Library_Management

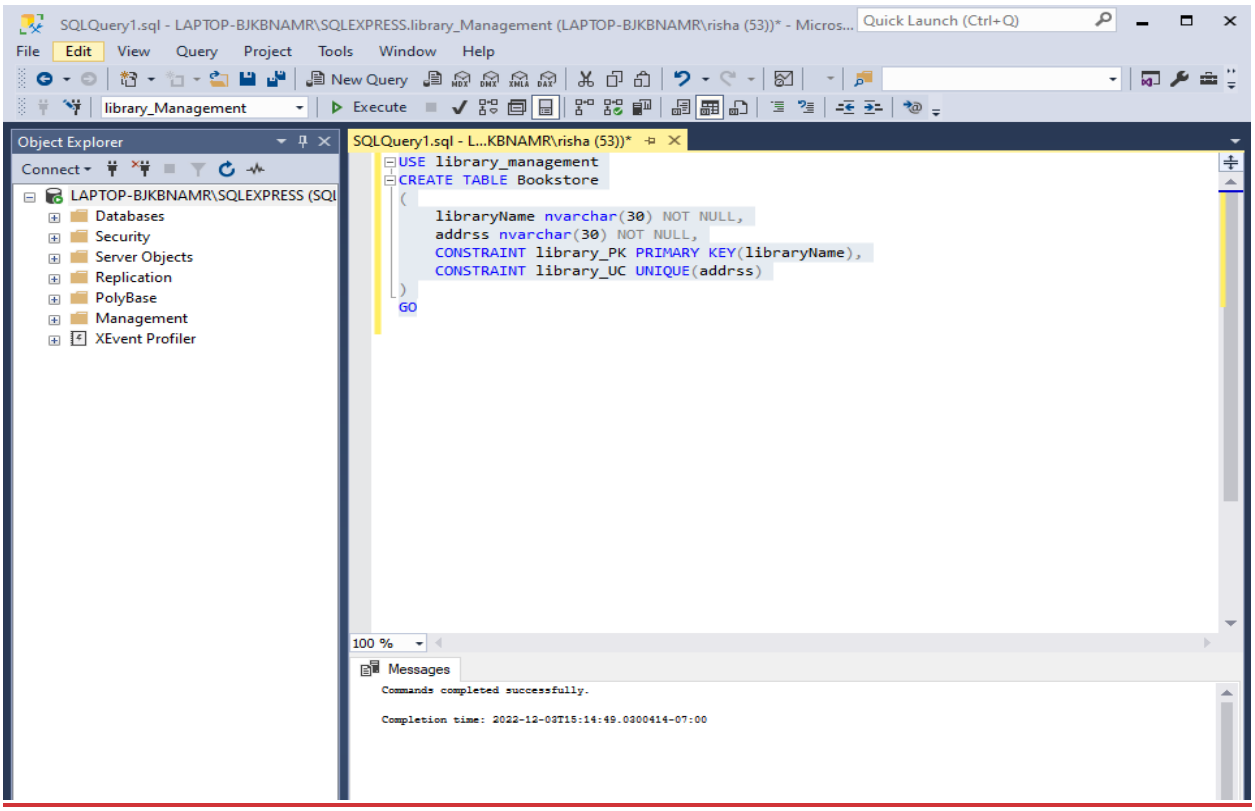
  SCREENSHOT:



CODE:

```
USE library_management

CREATE TABLE library

(

 libraryName nvarchar(30) NOT NULL,

        addrss nvarchar(30) NOT NULL,

        CONSTRAINT library_PK PRIMARY KEY(libraryName),

        CONSTRAINT library_UC UNIQUE(address)

)

GO
```

SCREENSHOT



- To alter table and add foreign key constraint:

  CODE:

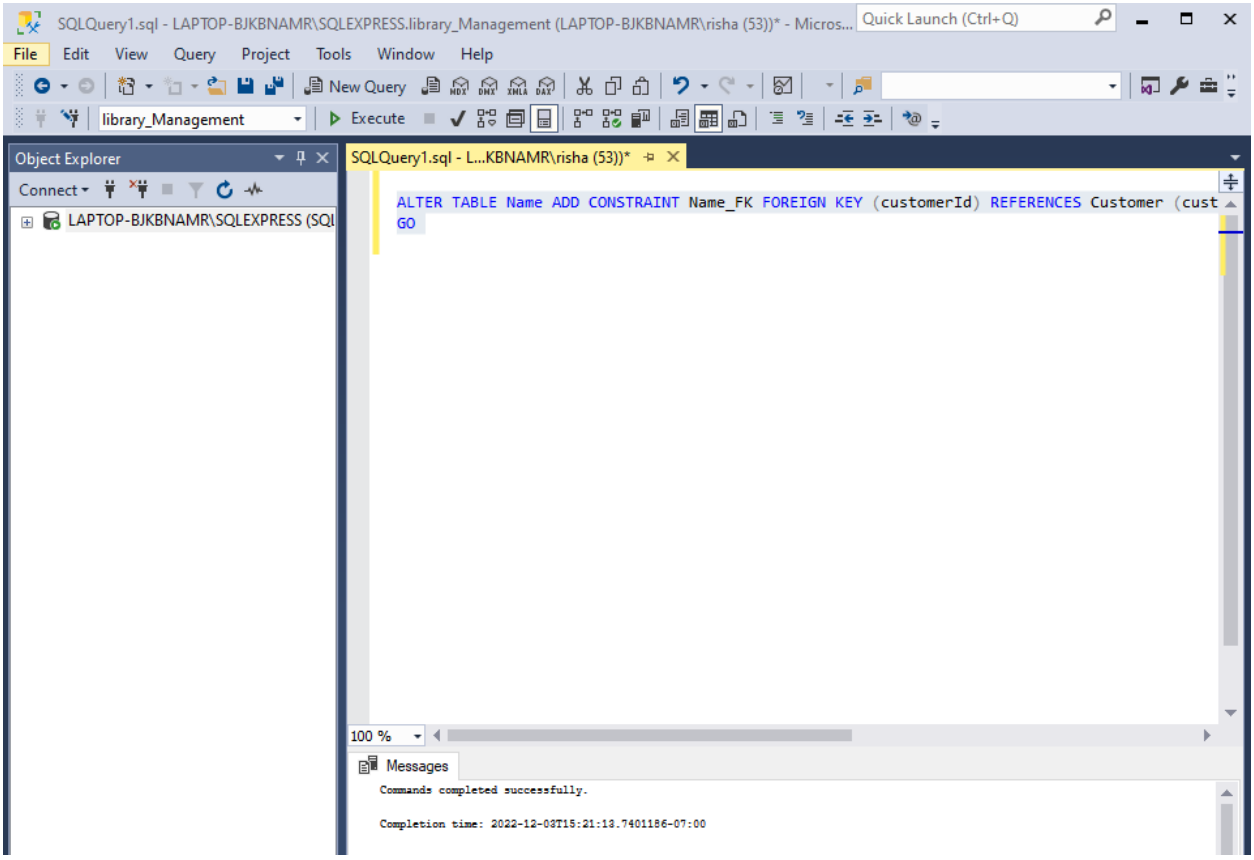
  ALTER TABLE Name ADD CONSTRAINT Name_FK FOREIGN KEY (customerId)

  REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO

  ACTION

  GO

SCREENSHOT:



- Inserting values to the Author table:

  CODE:

  Insert into Author (authorName)

values ('J.K Rowling'),

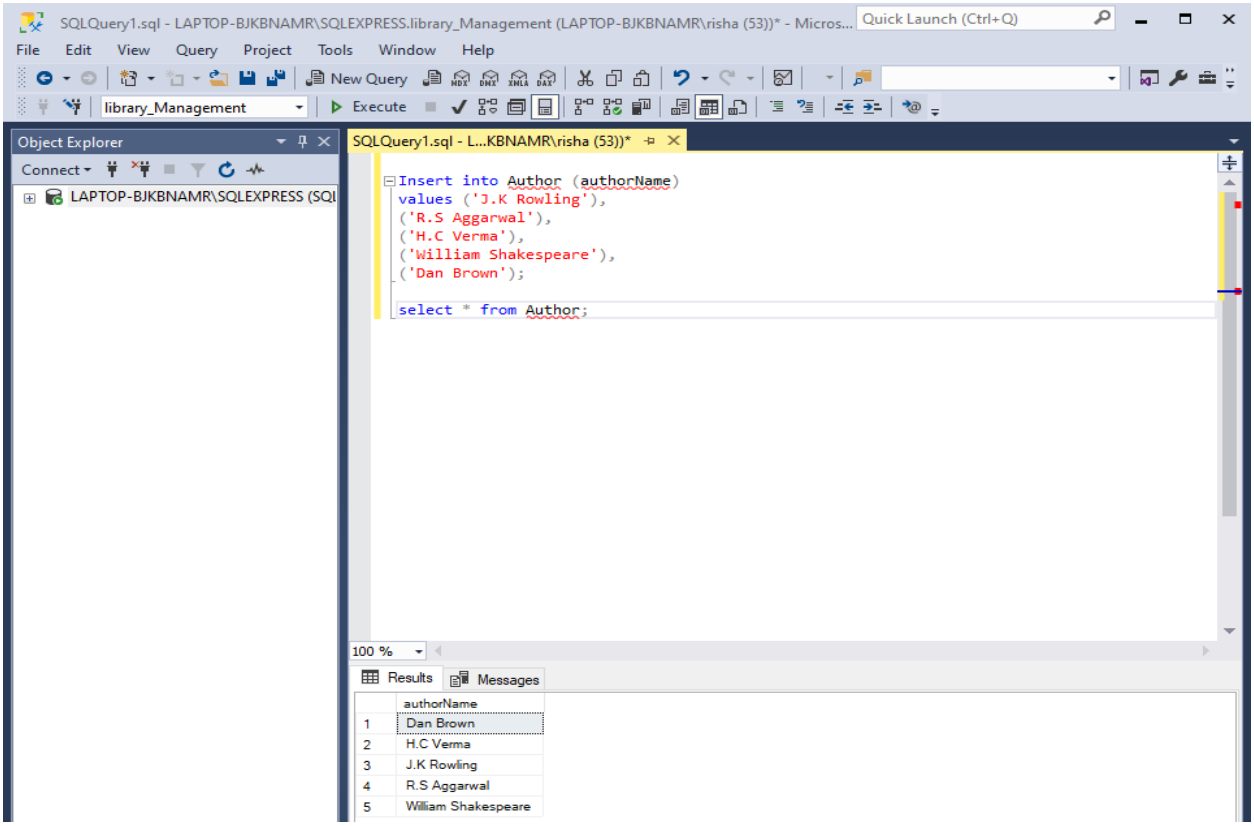
('R.S Aggarwal'),

('H.C Verma'),

('William Shakespeare'),

('Dan Brown');

SCREENSHOT:



- Displaying the values of library table which is in Tempe:

  CODE:

  SELECT * FROM library

  WHERE address=`Tempe`

  SCREENSHOT:

- Creating Trigger to display all the deleted item:

  CODE:

  CREATE TRIGGER DisplayDeletedStore

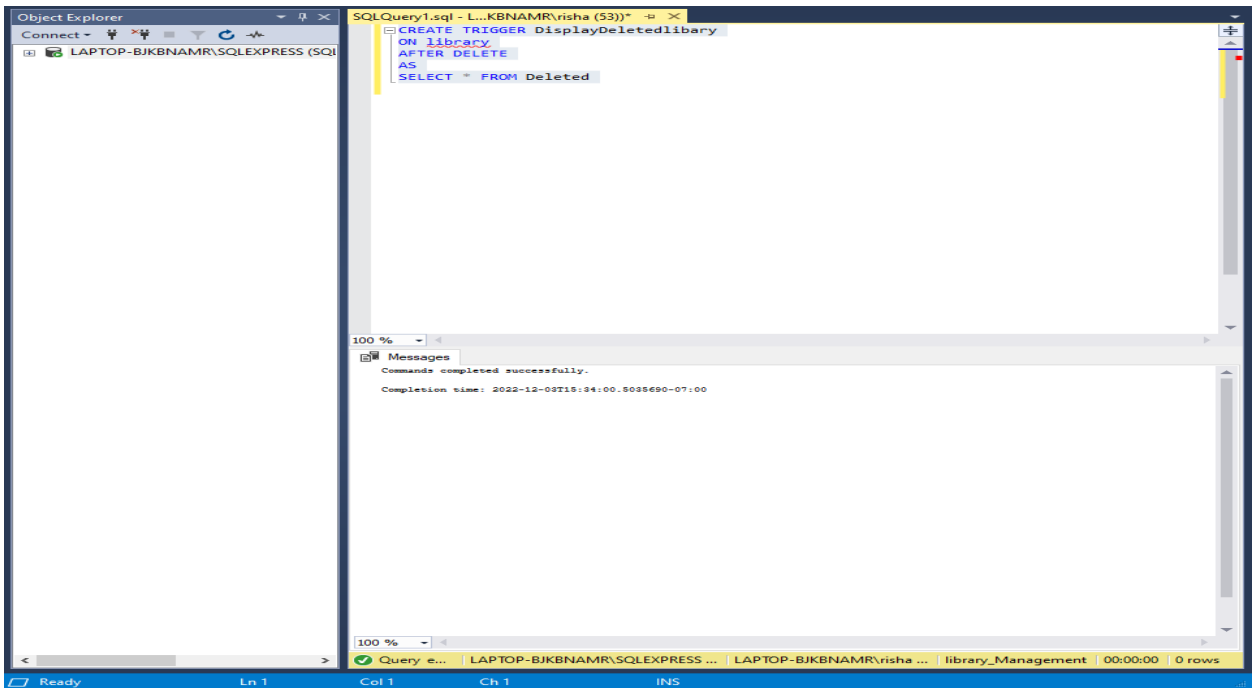
  ON library

  AFTER DELETE

  AS

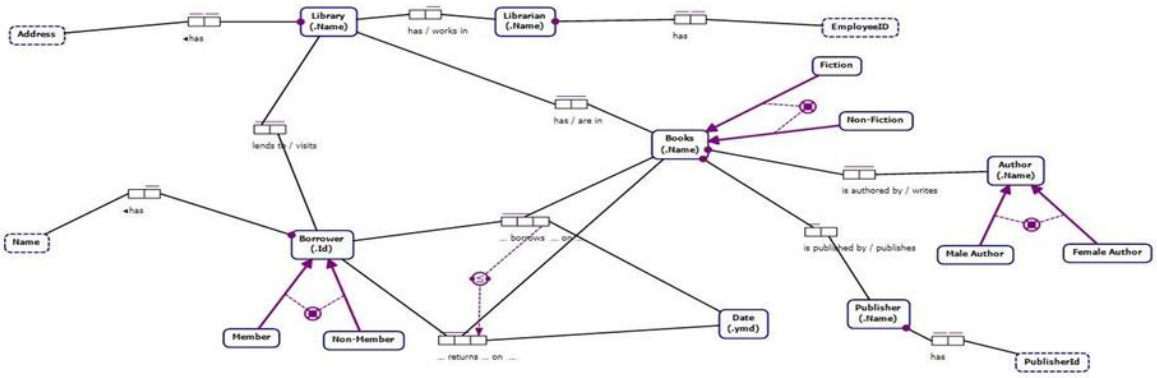
  SELECT * FROM Deleted

  SCREENSHOT:



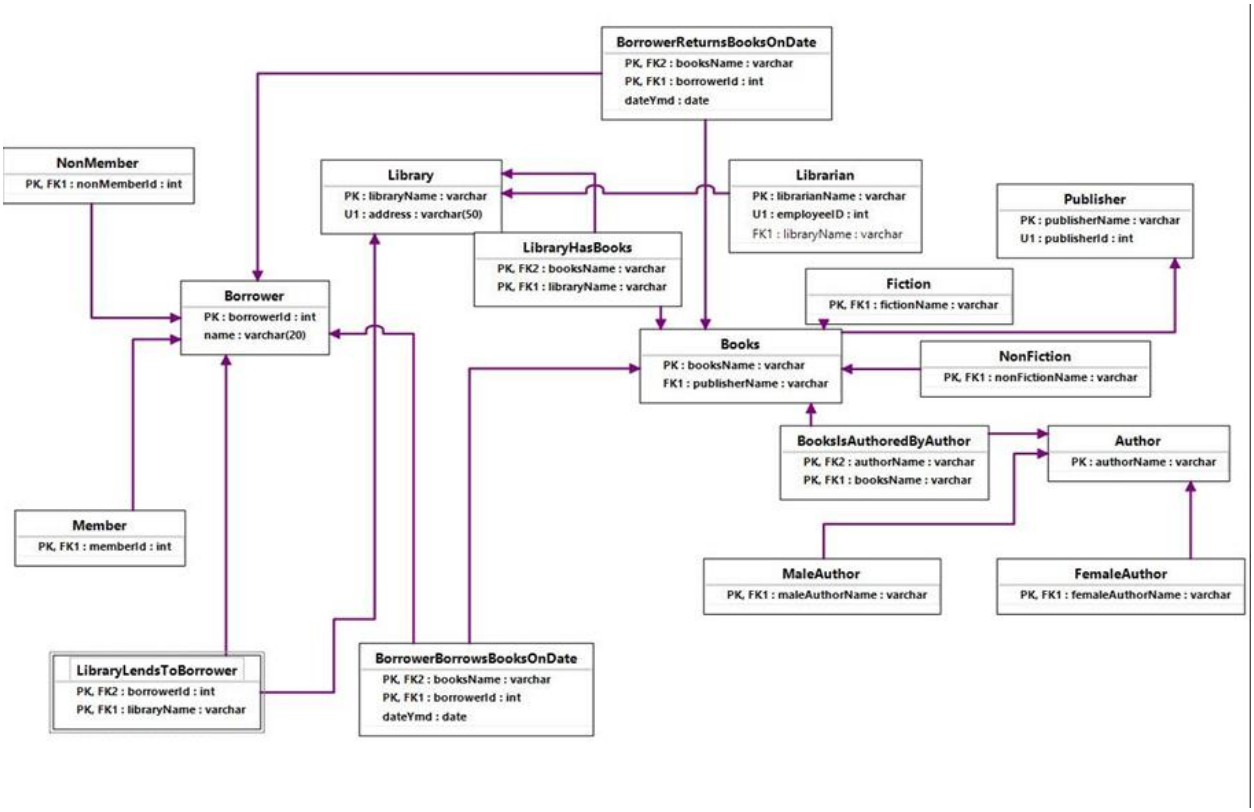
Section 2 - The ORM diagram

The project management tool for our library is depicted in the image below. As you can see, a library has been established with a name, an address, and subcategories for books, employees, and clients. Each subtype of an entity has its own subtypes; for example, an employee's name and employee ID, a customer's name, member status, and nonmember status, and a book's genre, author, publisher, and ID, as well as a link to a customer's account for dates to store.



Section 3– the Relational Schema

The relational View of our project is depicted in the image below. Relational view, which was taken from the ORM diagram above, has 18 tables, each of which is connected to the others through primary and foreign keys.



Generated SQL Code from ORM

CREATE SCHEMA ORMModel1

GO

GO

```
CREATE TABLE ORMModel1.library

(

	libraryName nvarchar(max) NOT NULL,

	lens nvarchar(20) NOT NULL,

	address nvarchar(20),

	CONSTRAINT library_UC1 UNIQUE(lens),

	CONSTRAINT library_PK PRIMARY KEY(libraryName)

)

GO



CREATE VIEW ORMModel1.library_UC2 (address)

WITH SCHEMABINDING

AS

SELECT address

FROM

ORMModel1.library
```

```
WHERE address IS NOT NULL

GO



CREATE UNIQUE CLUSTERED INDEX library_UC2Index ON

ORMModel1.library_UC2(address)

GO



CREATE TABLE ORMModel1.Employee

(

	employeeName nvarchar(20) NOT NULL,

	employeeID int IDENTITY (1, 1) NOT NULL,

	libraryName nvarchar(max),

	CONSTRAINT Employee_PK PRIMARY KEY(employeeName),

	CONSTRAINT Employee_UC UNIQUE(employeeID)

)

GO
```

```
CREATE TABLE ORMModel1.Customer
(
	customerId int IDENTITY (1, 1) NOT NULL,
	CONSTRAINT Customer_PK PRIMARY KEY(customerId)
)
GO

CREATE TABLE ORMModel1.Name
(
	"value" nvarchar(20) NOT NULL,
	customerId int NOT NULL,
	CONSTRAINT Name_PK PRIMARY KEY("value")
)
GO

CREATE TABLE ORMModel1.libraryLendsToCustomer
```

```
(
	libraryName nvarchar(max) NOT NULL,
	customerId int NOT NULL,
	CONSTRAINT libraryLendsToCustomer_PK PRIMARY KEY(libraryName,
customerId)
)
GO


CREATE TABLE ORMModel1.Books
(
	booksName nvarchar(20) NOT NULL,
	publisherName nvarchar(20) NOT NULL,
	CONSTRAINT Books_PK PRIMARY KEY(booksName)
)
GO
```

```
CREATE TABLE ORMModel1.libraryHasBooks
(
	booksName nvarchar(20) NOT NULL,
	libraryName nvarchar(max) NOT NULL,
	CONSTRAINT libraryHasBooks_PK PRIMARY KEY(libraryName, booksName)
)
GO

CREATE TABLE ORMModel1.Fiction
(
	fictionName nvarchar(20) NOT NULL,
	CONSTRAINT Fiction_PK PRIMARY KEY(fictionName)
)
GO

CREATE TABLE ORMModel1.NonFiction
(
```

```
    nonFictionName nvarchar(20) NOT NULL,
    CONSTRAINT NonFiction_PK PRIMARY KEY(nonFictionName)
)
GO

CREATE TABLE ORMModel1.Publisher
(
    publisherName nvarchar(20) NOT NULL,
    publisherId nvarchar(20) NOT NULL,
    CONSTRAINT Publisher_PK PRIMARY KEY(publisherName),
    CONSTRAINT Publisher_UC UNIQUE(publisherId)
)
GO

CREATE TABLE ORMModel1.Author
(
    authorName nvarchar(20) NOT NULL,
```

```
    CONSTRAINT Author_PK PRIMARY KEY(authorName)
)
GO

CREATE TABLE ORMModel1.BooksIsAuthoredByAuthor
(
    authorName nvarchar(20) NOT NULL,
    booksName nvarchar(20) NOT NULL,
    CONSTRAINT BooksIsAuthoredByAuthor_PK PRIMARY KEY(booksName,
authorName)
)
GO

CREATE TABLE ORMModel1.MaleAUthor
(
    maleAUthorName nvarchar(20) NOT NULL,
```

```
    CONSTRAINT MaleAUthor_PK PRIMARY KEY(maleAUthorName)
)
GO

CREATE TABLE ORMModel1.FemaleAuthor
(
    femaleAuthorName nvarchar(20) NOT NULL,
    CONSTRAINT FemaleAuthor_PK PRIMARY KEY(femaleAuthorName)
)
GO

CREATE TABLE ORMModel1."Member"
(
    memberId int NOT NULL,
    CONSTRAINT Member_PK PRIMARY KEY(memberId)
)
GO
```

```
CREATE TABLE ORMModel1.NonMember
(
	nonMemberId int NOT NULL,
	CONSTRAINT NonMember_PK PRIMARY KEY(nonMemberId)
)
GO

CREATE TABLE ORMModel1.CustomerBorrowsDataeOnBooks
(
	customerId int NOT NULL,
	datae nvarchar(20) NOT NULL,
	booksName nvarchar(20) NOT NULL,
	CONSTRAINT CustomerBorrowsDataeOnBooks_PK PRIMARY KEY(customerId,
datae)
)
GO
```

```
CREATE TABLE ORMModel1.CustomerReturnsDataeOnBooks
(
	customerId int NOT NULL,
	datae nvarchar(20) NOT NULL,
	booksName nvarchar(20) NOT NULL,
	CONSTRAINT CustomerReturnsDataeOnBooks_PK PRIMARY KEY(customerId, datae)
)
GO
ALTER TABLE ORMModel1.Employee ADD CONSTRAINT Employee_FK FOREIGN KEY (libraryName) REFERENCES ORMModel1.library (libraryName) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.Name ADD CONSTRAINT Name_FK FOREIGN KEY (customerId) REFERENCES ORMModel1.Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

```
ALTER TABLE ORMModel1.libraryLendsToCustomer ADD CONSTRAINT
libraryLendsToCustomer_FK1 FOREIGN KEY (libraryName) REFERENCES
ORMModel1.library (libraryName) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.libraryLendsToCustomer ADD CONSTRAINT
libraryLendsToCustomer_FK2 FOREIGN KEY (customerId) REFERENCES
ORMModel1.Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.Books ADD CONSTRAINT Books_FK FOREIGN KEY
(publisherName) REFERENCES ORMModel1.Publisher (publisherName) ON DELETE NO
ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.libraryHasBooks ADD CONSTRAINT libraryHasBooks_FK1
FOREIGN KEY (libraryName) REFERENCES ORMModel1.library (libraryName) ON
DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.libraryHasBooks ADD CONSTRAINT libraryHasBooks_FK2
FOREIGN KEY (booksName) REFERENCES ORMModel1.Books (booksName) ON DELETE
NO ACTION ON UPDATE NO ACTION

GO
```

```
ALTER TABLE ORMModel1.Fiction ADD CONSTRAINT Fiction_FK FOREIGN KEY
(fictionName) REFERENCES ORMModel1.Books (booksName) ON DELETE NO ACTION
ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.NonFiction ADD CONSTRAINT NonFiction_FK FOREIGN
KEY (nonFictionName) REFERENCES ORMModel1.Books (booksName) ON DELETE NO
ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.BooksIsAuthoredByAuthor ADD CONSTRAINT
BooksIsAuthoredByAuthor_FK1 FOREIGN KEY (booksName) REFERENCES
ORMModel1.Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.BooksIsAuthoredByAuthor ADD CONSTRAINT
BooksIsAuthoredByAuthor_FK2 FOREIGN KEY (authorName) REFERENCES
ORMModel1.Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.MaleAUthor ADD CONSTRAINT MaleAUthor_FK FOREIGN
KEY (maleAUthorName) REFERENCES ORMModel1.Author (authorName) ON DELETE NO
ACTION ON UPDATE NO ACTION

GO
```

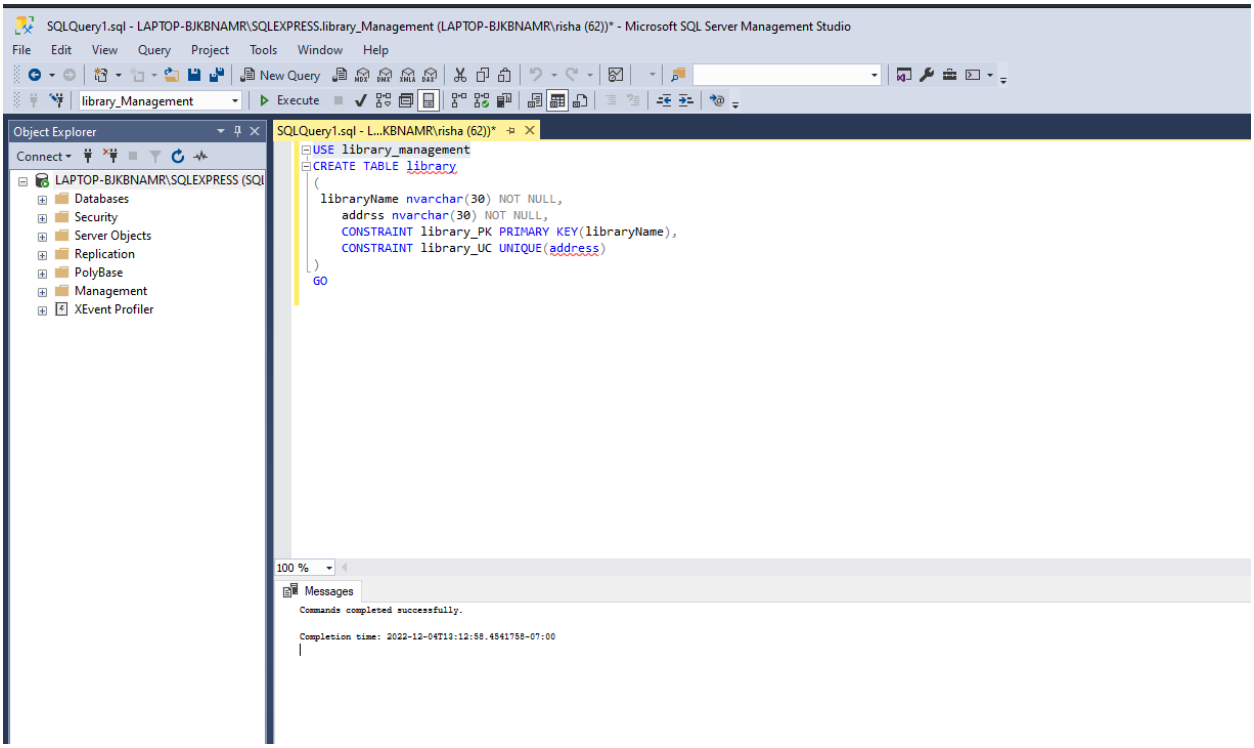
```
ALTER TABLE ORMModel1.FemaleAuthor ADD CONSTRAINT FemaleAuthor_FK
FOREIGN KEY (femaleAuthorName) REFERENCES ORMModel1.Author (authorName) ON
DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1."Member" ADD CONSTRAINT Member_FK FOREIGN KEY
(memberId) REFERENCES ORMModel1.Customer (customerId) ON DELETE NO ACTION
ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.NonMember ADD CONSTRAINT NonMember_FK FOREIGN
KEY (nonMemberId) REFERENCES ORMModel1.Customer (customerId) ON DELETE NO
ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.CustomerBorrowsDataeOnBooks ADD CONSTRAINT
CustomerBorrowsDataeOnBooks_FK1 FOREIGN KEY (customerId) REFERENCES
ORMModel1.Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.CustomerBorrowsDataeOnBooks ADD CONSTRAINT
CustomerBorrowsDataeOnBooks_FK2 FOREIGN KEY (booksName) REFERENCES
ORMModel1.Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION

GO
```

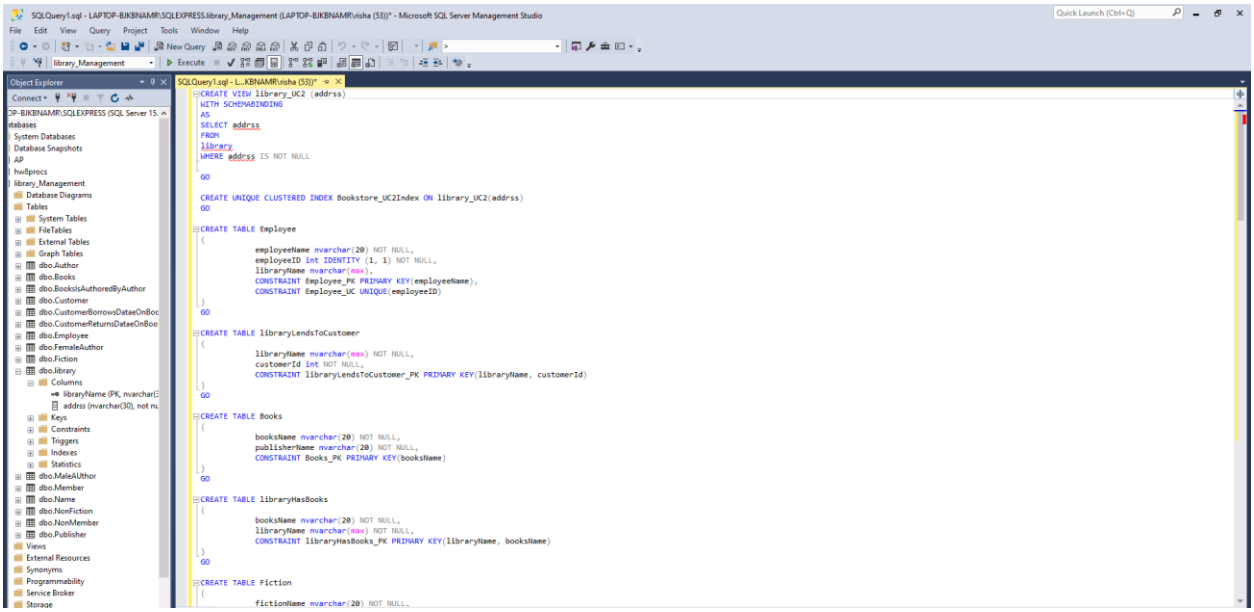
```
ALTER TABLE ORMModel1.CustomerReturnsDataeOnBooks ADD CONSTRAINT
CustomerReturnsDataeOnBooks_FK1 FOREIGN KEY (customerId) REFERENCES
ORMModel1.Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

ALTER TABLE ORMModel1.CustomerReturnsDataeOnBooks ADD CONSTRAINT
CustomerReturnsDataeOnBooks_FK2 FOREIGN KEY (booksName) REFERENCES
ORMModel1.Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

GO
```

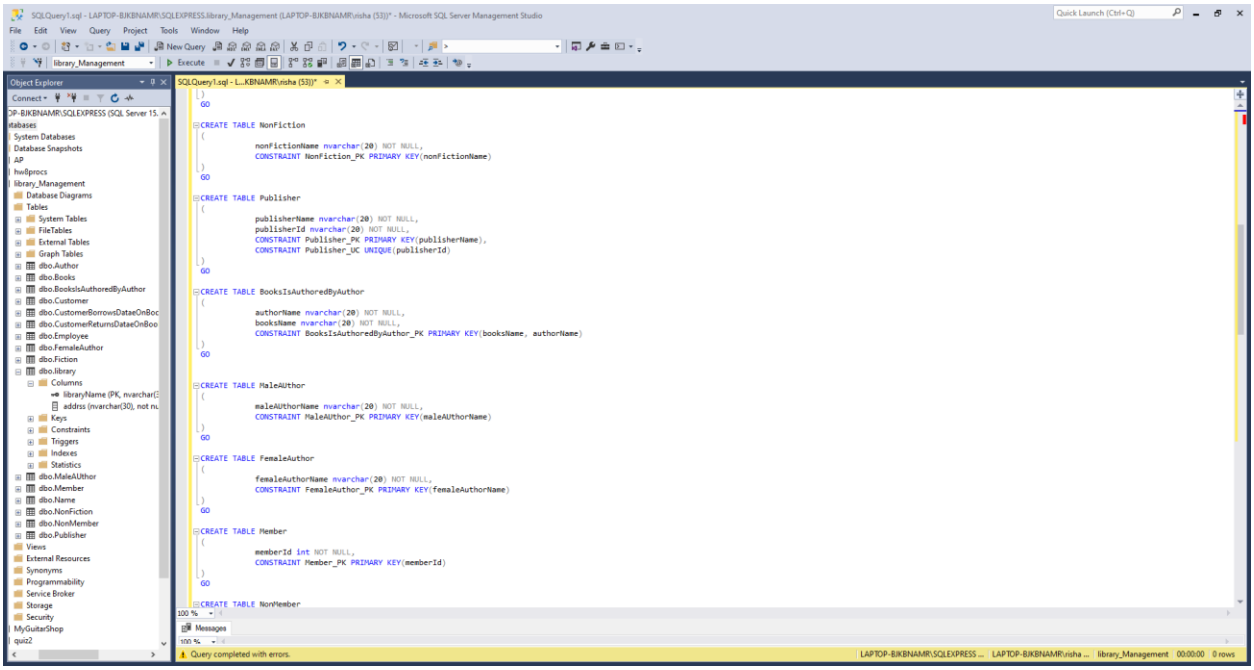
Section 4

Creating Tables and Column Constraints using DDL commonds:

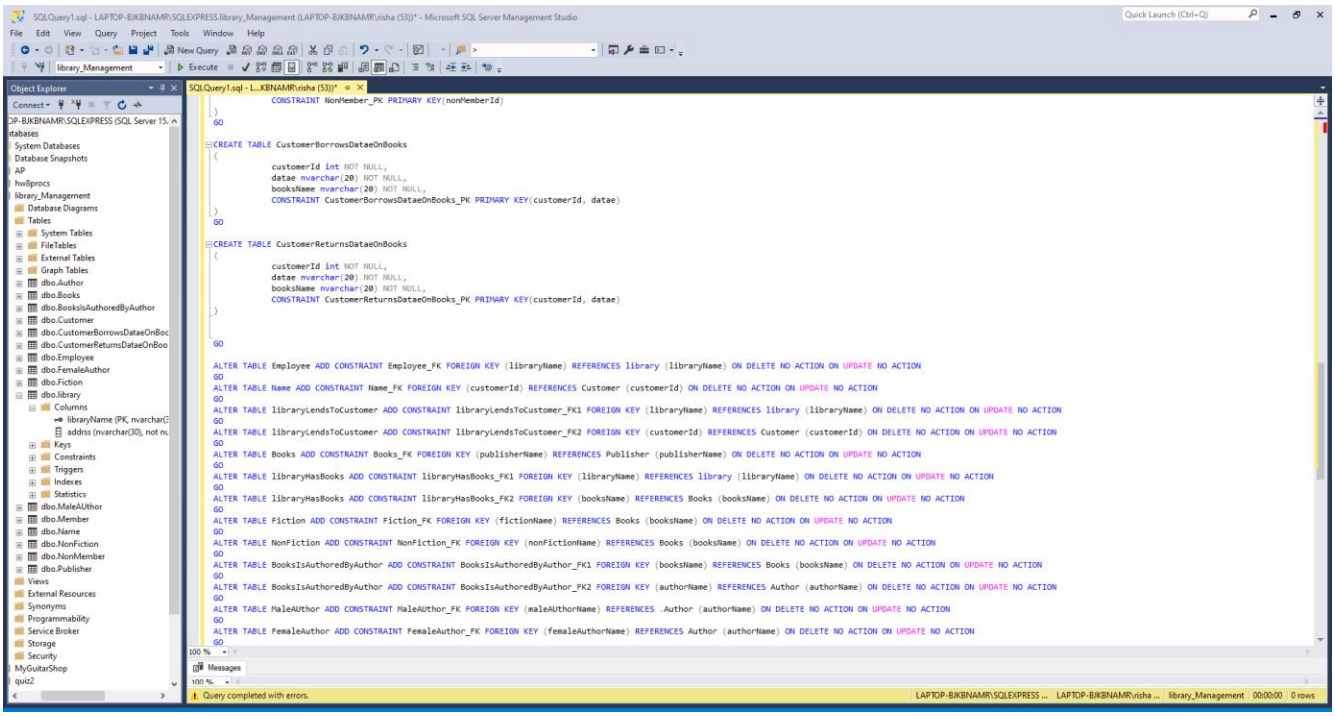
In order to meet the requirements of the table and column constraints, respectively, we applied the UNIQUE and NOT NULL constraints.

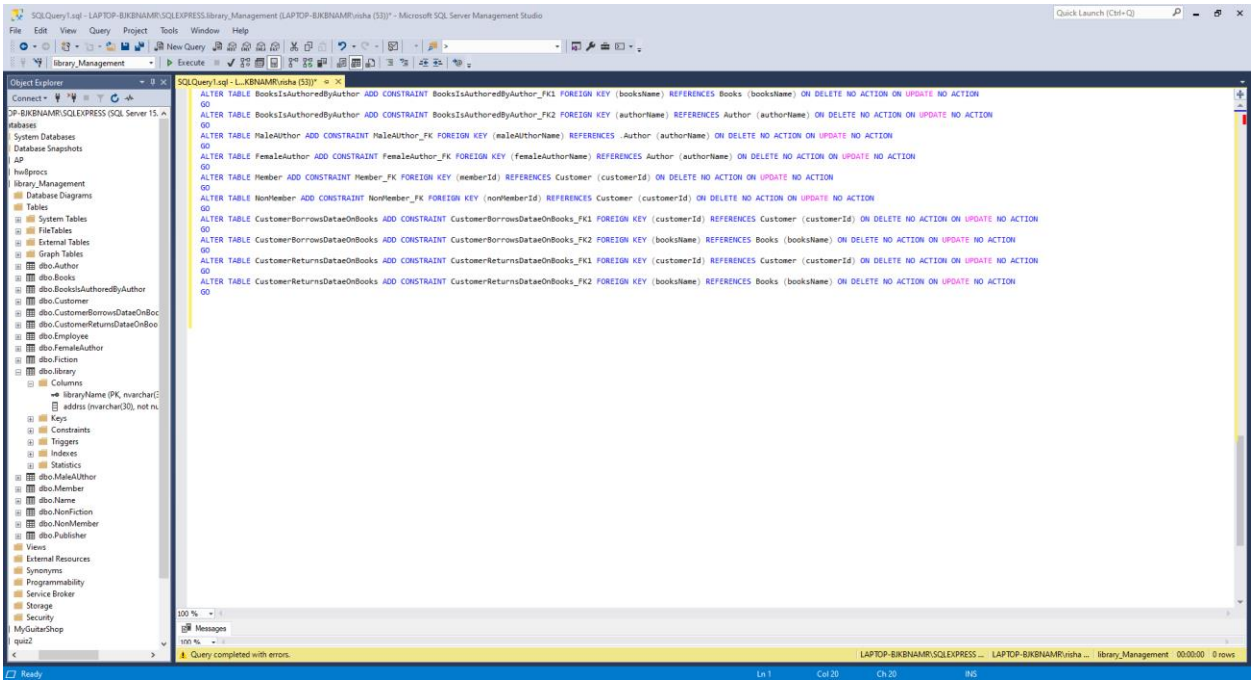




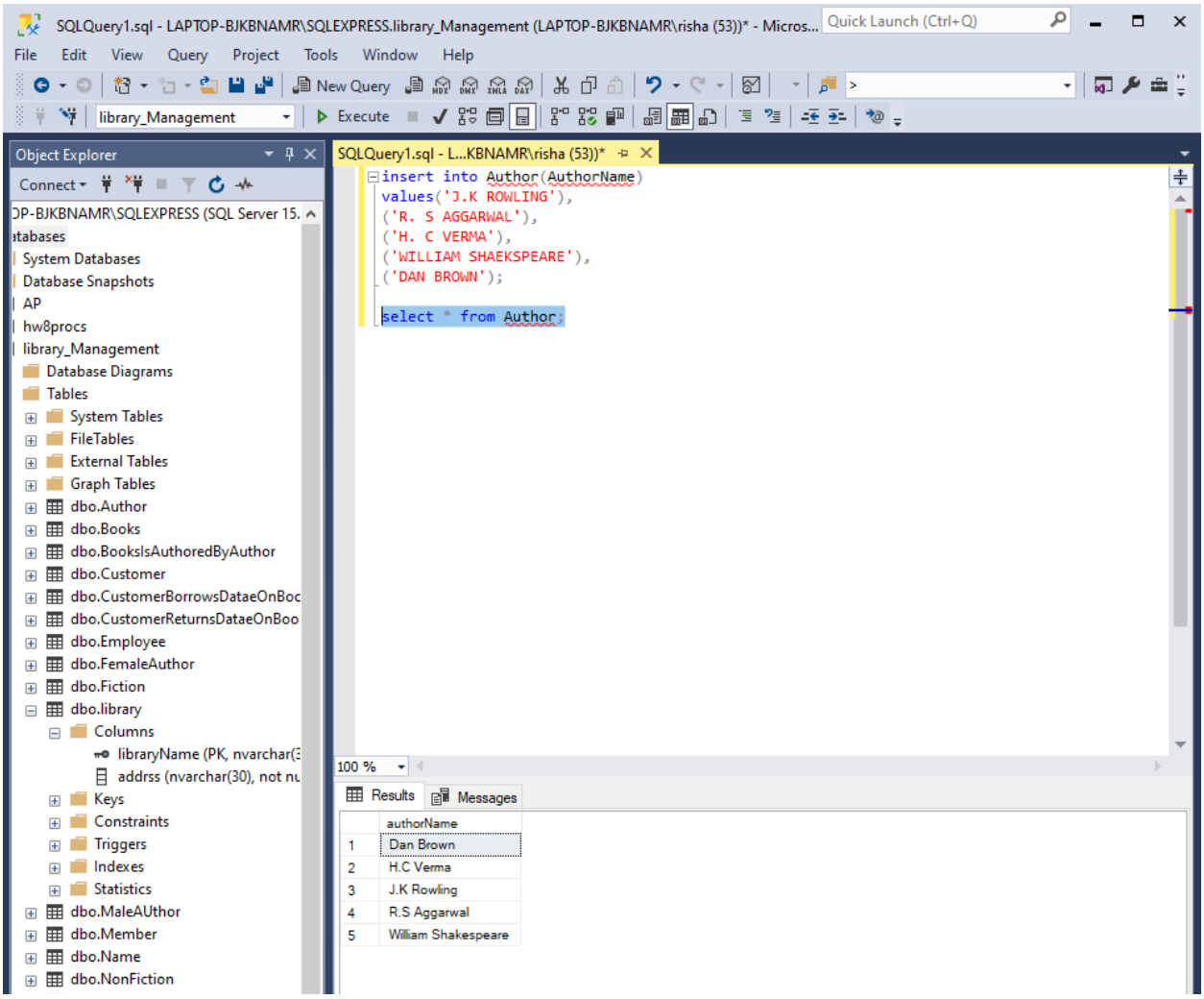


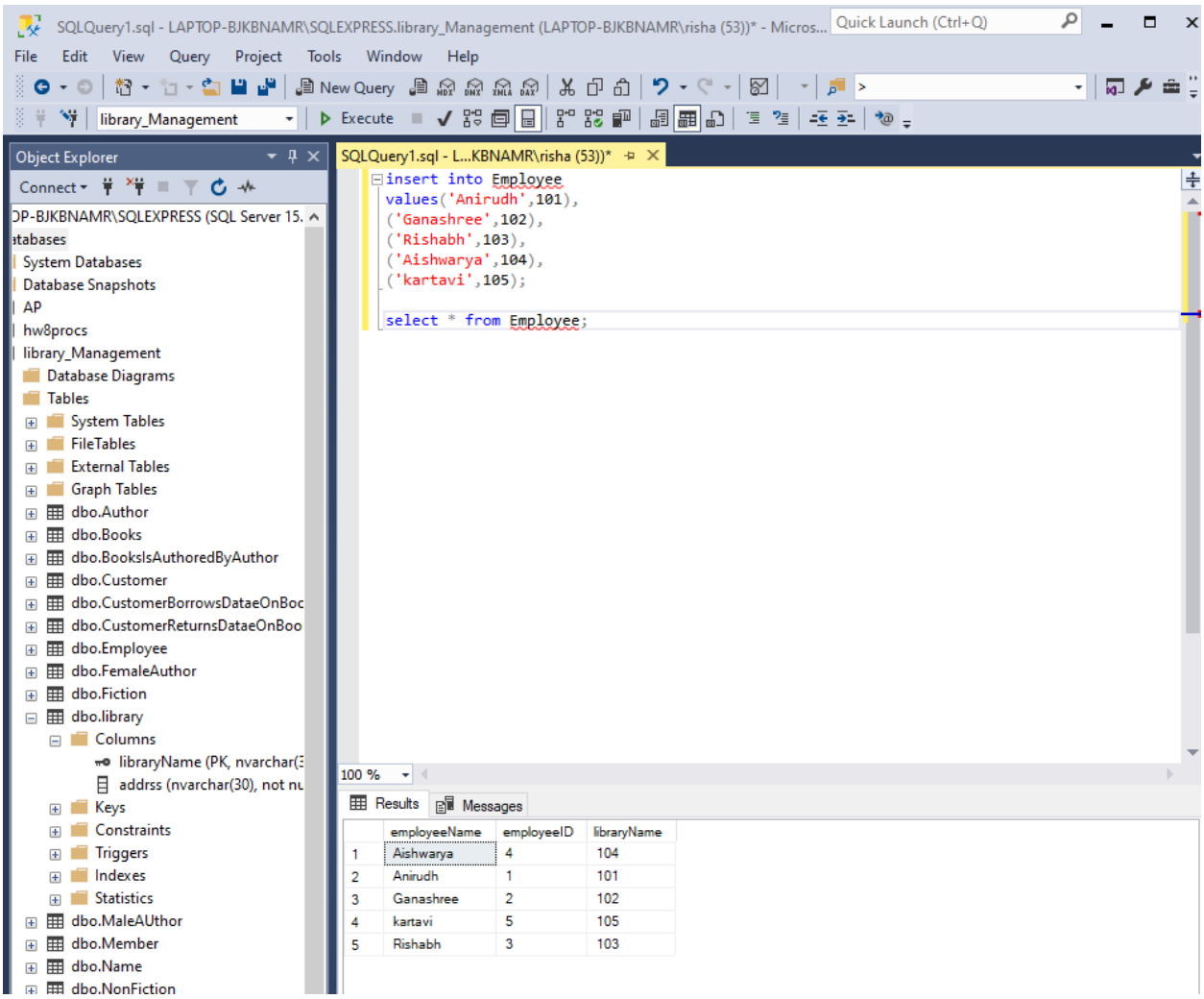
Foriegn Key constraints:

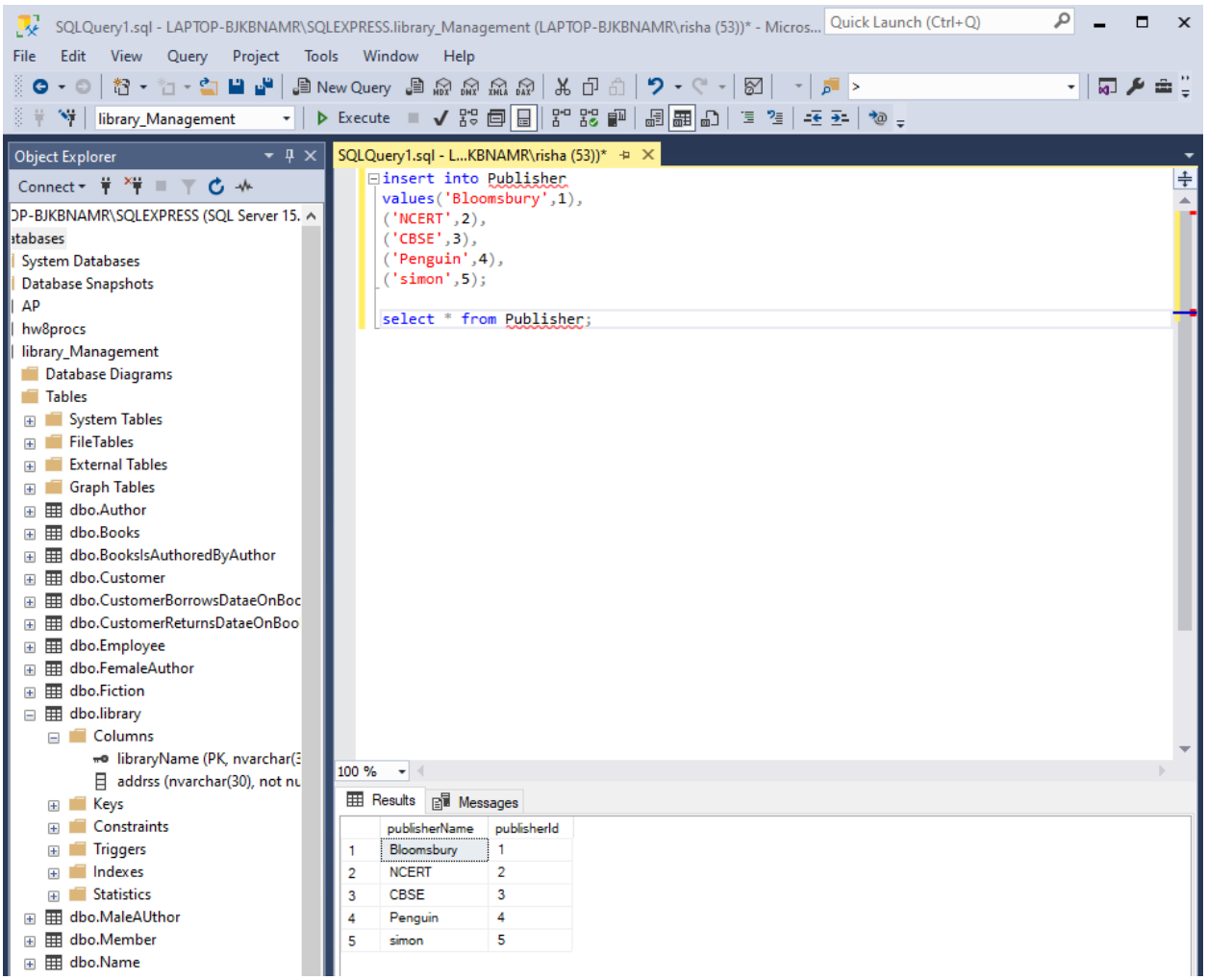

SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Microsoft SQL Server Management Studio
CONSTRAINT NonMember_PK PRIMARY KEY(nonMemberId)
GO
CREATE TABLE CustomerBorrowsDataeOnBooks
customerId int NOT NULL,
datae nvarchar(20) NOT NULL,
booksName nvarchar(20) NOT NULL,
CONSTRAINT CustomerBorrowsDataeOnBooks_PK PRIMARY KEY(customerId, datae)
CREATE TABLE CustomerReturnsDataeOnBooks
customerId int NOT NULL,
datae nvarchar(20) NOT NULL,
booksName nvarchar(20) NOT NULL,
CONSTRAINT CustomerReturnsDataeOnBooks_PK PRIMARY KEY(customerId, datae)
ALTER TABLE Employee ADD CONSTRAINT Employee_FK FOREIGN KEY (libraryName) REFERENCES library (libraryName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE Name ADD CONSTRAINT Name_FK FOREIGN KEY (customerId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE libraryLendsToCustomer ADD CONSTRAINT libraryLendsToCustomer_FK1 FOREIGN KEY (libraryName) REFERENCES library (libraryName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE libraryLendsToCustomer ADD CONSTRAINT libraryLendsToCustomer_FK2 FOREIGN KEY (customerId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE Books ADD CONSTRAINT Books_FK FOREIGN KEY (publisherName) REFERENCES Publisher (publisherName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE libraryHasBooks ADD CONSTRAINT libraryHasBooks_FK1 FOREIGN KEY (libraryName) REFERENCES library (libraryName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE libraryHasBooks ADD CONSTRAINT libraryHasBooks_FK2 FOREIGN KEY (booksName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE Fiction ADD CONSTRAINT Fiction_FK FOREIGN KEY (fictionName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE NonFiction ADD CONSTRAINT NonFiction_FK FOREIGN KEY (nonFictionName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE BooksIsAuthoredByAuthor ADD CONSTRAINT BooksIsAuthoredByAuthor_FK1 FOREIGN KEY (booksName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE BooksIsAuthoredByAuthor ADD CONSTRAINT BooksIsAuthoredByAuthor_FK2 FOREIGN KEY (authorName) REFERENCES Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE MaleAUthor ADD CONSTRAINT MaleAUthor_FK FOREIGN KEY (maleAUthorName) REFERENCES .Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE FemaleAuthor ADD CONSTRAINT FemaleAuthor_FK FOREIGN KEY (femaleAuthorName) REFERENCES Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
Query completed with errors.

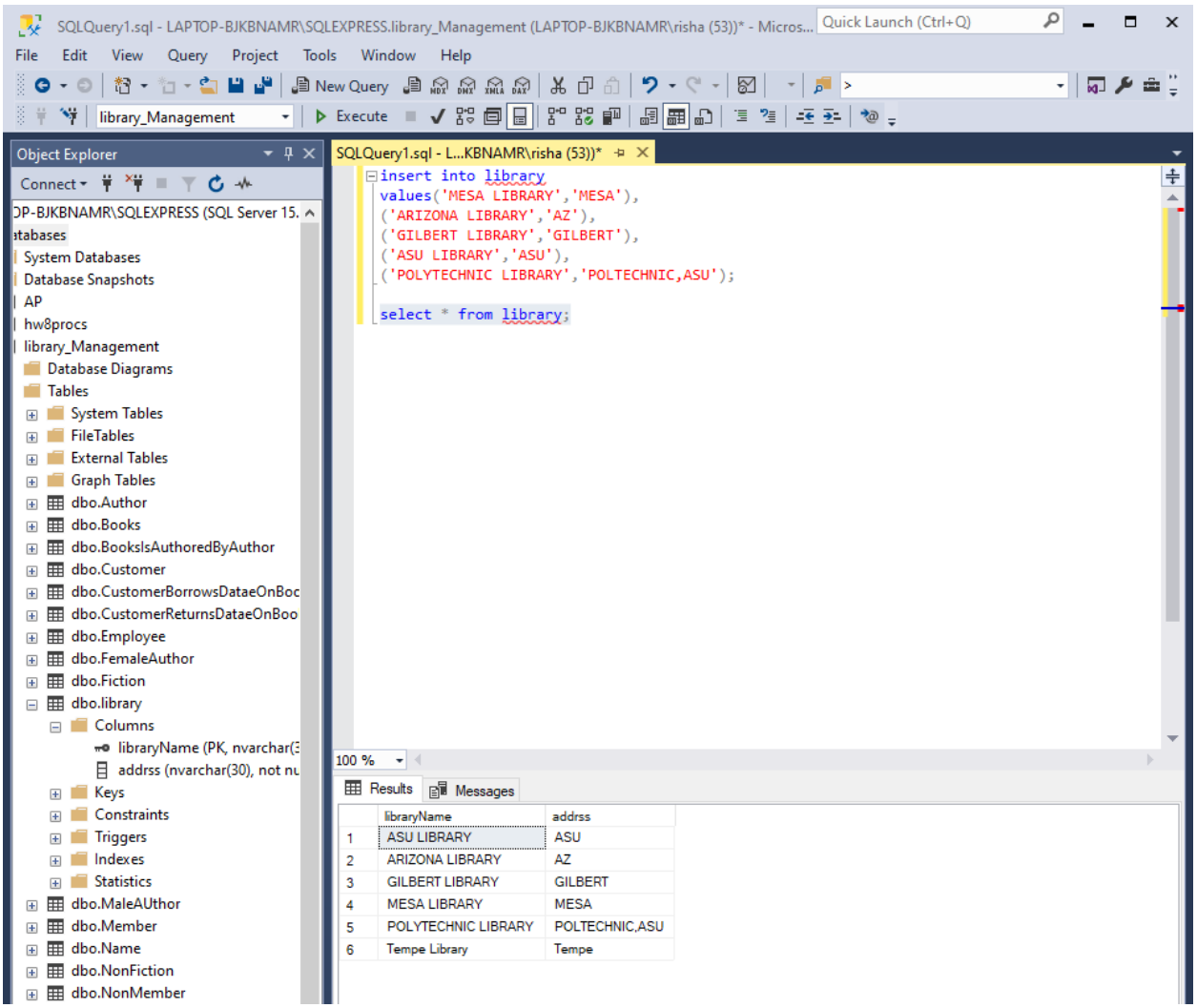

SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Microsoft SQL Server Management Studio
ALTER TABLE BooksIsAuthoredByAuthor ADD CONSTRAINT BooksIsAuthoredByAuthor_FK1 FOREIGN KEY (booksName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE BooksIsAuthoredByAuthor ADD CONSTRAINT BooksIsAuthoredByAuthor_FK2 FOREIGN KEY (authorName) REFERENCES Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE MaleAUthor ADD CONSTRAINT MaleAUthor_FK FOREIGN KEY (maleAUthorName) REFERENCES .Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE FemaleAuthor ADD CONSTRAINT FemaleAuthor_FK FOREIGN KEY (femaleAuthorName) REFERENCES Author (authorName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE Member ADD CONSTRAINT Member_FK FOREIGN KEY (memberId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE NonMember ADD CONSTRAINT NonMember_FK FOREIGN KEY (nonMemberId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE CustomerBorrowsDataeOnBooks ADD CONSTRAINT CustomerBorrowsDataeOnBooks_FK1 FOREIGN KEY (customerId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE CustomerBorrowsDataeOnBooks ADD CONSTRAINT CustomerBorrowsDataeOnBooks_FK2 FOREIGN KEY (booksName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE CustomerReturnsDataeOnBooks ADD CONSTRAINT CustomerReturnsDataeOnBooks_FK1 FOREIGN KEY (customerId) REFERENCES Customer (customerId) ON DELETE NO ACTION ON UPDATE NO ACTION
ALTER TABLE CustomerReturnsDataeOnBooks ADD CONSTRAINT CustomerReturnsDataeOnBooks_FK2 FOREIGN KEY (booksName) REFERENCES Books (booksName) ON DELETE NO ACTION ON UPDATE NO ACTION
Query completed with errors.
Ready

Adding Tables:

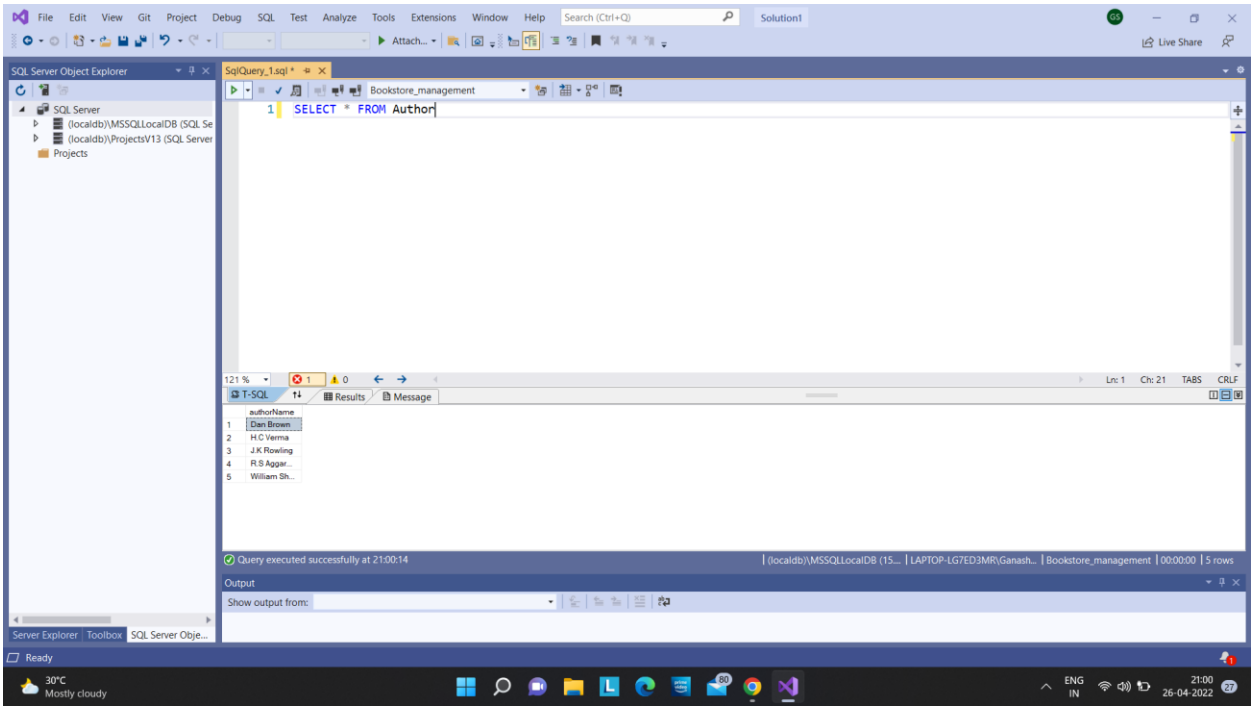


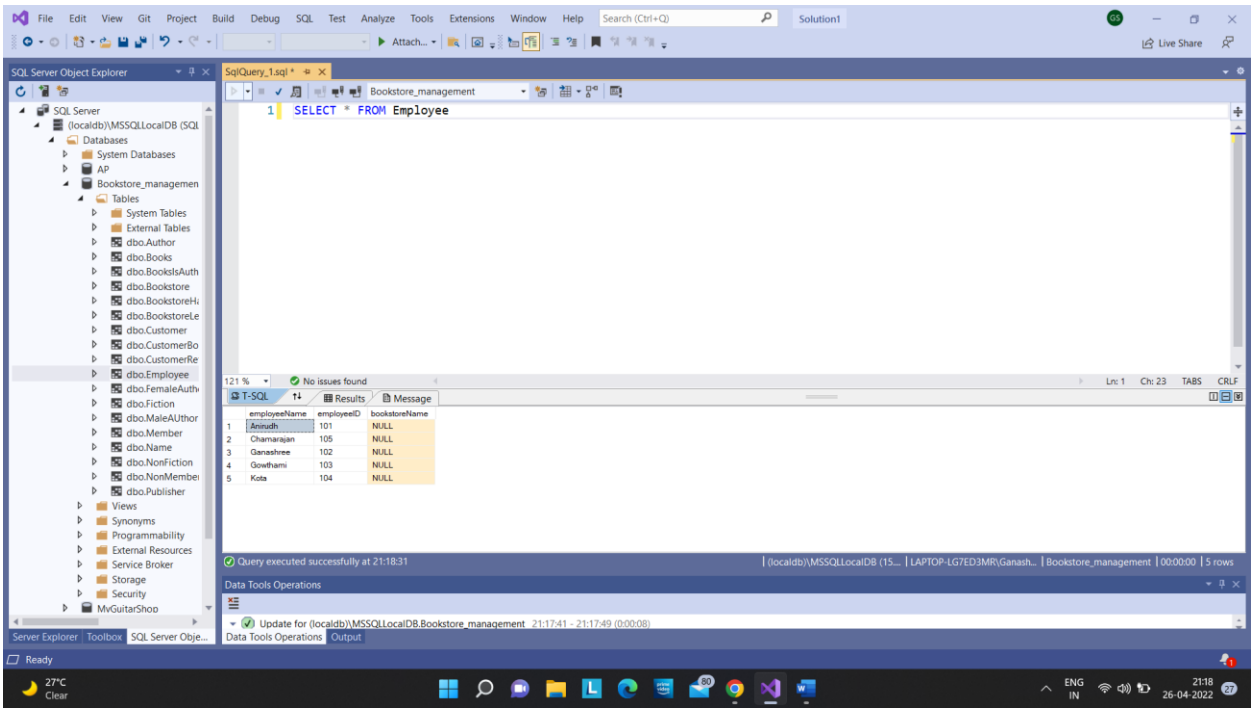

SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Micros...
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
library_Management
Execute
Object Explorer
Connect
DP-BJKBNAMR\SQLEXPRESS (SQL Server 15.
atabases
System Databases
Database Snapshots
AP
hw8procs
library_Management
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Author
dbo.Books
dbo.BooksIsAuthoredByAuthor
dbo.Customer
dbo.CustomerBorrowsDataeOnBoc
dbo.CustomerReturnsDataeOnBoo
dbo.Employee
dbo.FemaleAuthor
dbo.Fiction
dbo.library
Columns
libraryName (PK, nvarchar(3
addrss (nvarchar(30), not nu
Keys
Constraints
Triggers
Indexes
Statistics
dbo.MaleAUthor
dbo.Member
dbo.Name
dbo.NonFiction
SQLQuery1.sql - L...KBNAMR\risha (53))*
insert into Employee
values('Anirudh',101),
('Ganashree',102),
('Rishabh',103),
('Aishwarya',104),
('kartavi',105);
select * from Employee;
100 %
Results
Messages
employeeName employeeID libraryName
1 Aishwarya 4 104
2 Anirudh 1 101
3 Ganashree 2 102
4 kartavi 5 105
5 Rishabh 3 103

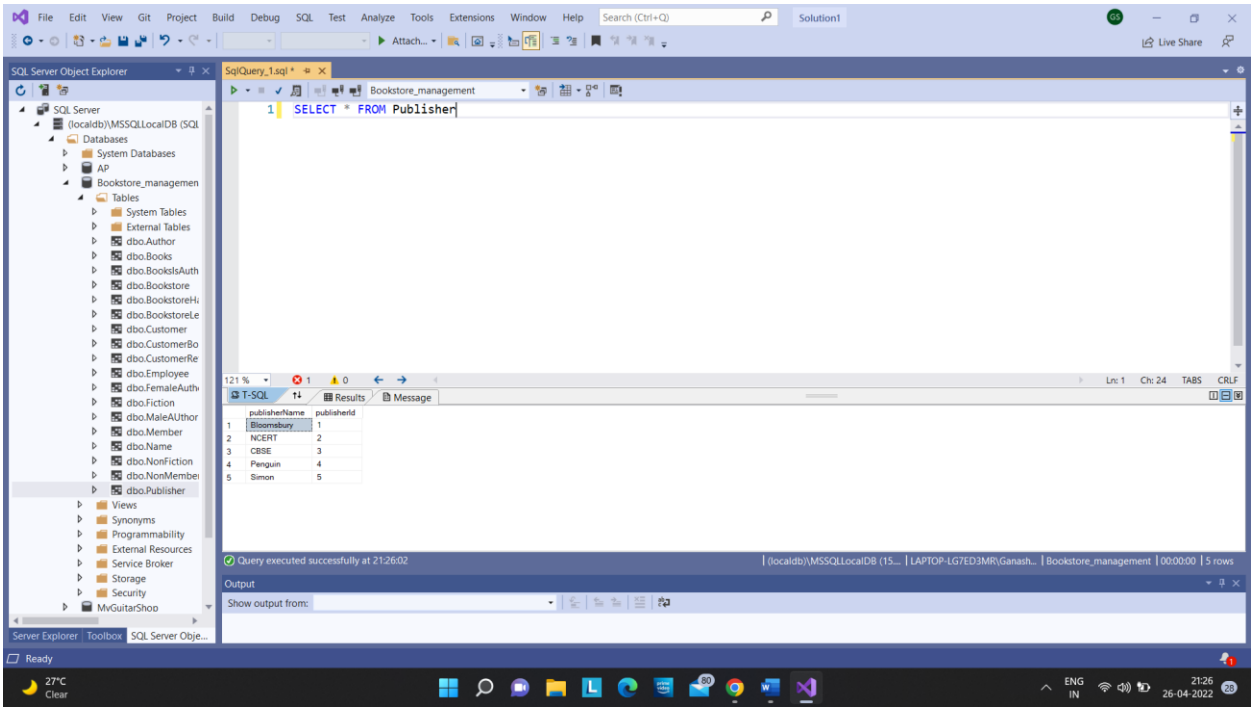

SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Micros...
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
library_Management
Execute
Object Explorer
Connect
SQLQuery1.sql - L...KBNAMR\risha (53))*
insert into Publisher
values('Bloomsbury',1),
('NCERT',2),
('CBSE',3),
('Penguin',4),
('simon',5);
select * from Publisher;
100 %
Results
Messages
publisherName
publisherId
1 Bloomsbury 1
2 NCERT 2
3 CBSE 3
4 Penguin 4
5 simon 5
System Databases
Database Snapshots
AP
hw8procs
library_Management
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Author
dbo.Books
dbo.BooksIsAuthoredByAuthor
dbo.Customer
dbo.Employee
dbo.FemaleAuthor
dbo.Fiction
dbo.library
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.MaleAUthor
dbo.Member
dbo.Name

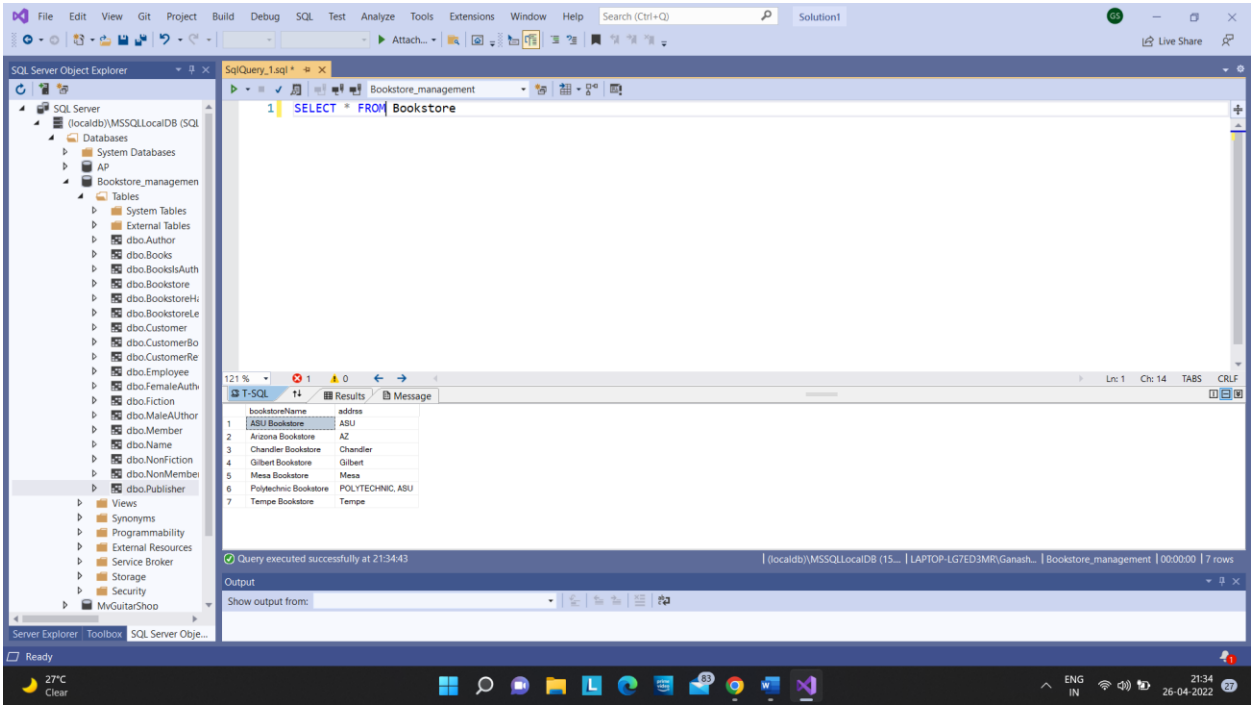

SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Micros...
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
library_Management
Execute
Object Explorer
Connect
OP-BJKBNAMR\SQLEXPRESS (SQL Server 15.
tabases
System Databases
Database Snapshots
AP
hw8procs
library_Management
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Author
dbo.Books
dbo.BooksIsAuthoredByAuthor
dbo.Customer
dbo.CustomerBorrowsDataeOnBoc
dbo.CustomerReturnsDataeOnBoo
dbo.Employee
dbo.FemaleAuthor
dbo.Fiction
dbo.library
Columns
libraryName (PK, nvarchar(
addrss (nvarchar(30), not nu
Keys
Constraints
Triggers
Indexes
Statistics
dbo.MaleAUthor
dbo.Member
dbo.Name
dbo.NonFiction
dbo.NonMember
SQLQuery1.sql - L...KBNAMR\risha (53))*
insert into library
values('MESA LIBRARY','MESA'),
('ARIZONA LIBRARY','AZ'),
('GILBERT LIBRARY','GILBERT'),
('ASU LIBRARY','ASU'),
('POLYTECHNIC LIBRARY','POLTECHNIC,ASU');
select * from library;
100 %
Results
Messages
libraryName addrss
1 ASU LIBRARY ASU
2 ARIZONA LIBRARY AZ
3 GILBERT LIBRARY GILBERT
4 MESA LIBRARY MESA
5 POLYTECHNIC LIBRARY POLTECHNIC,ASU
6 Tempe Library Tempe

Using DML commands to show the tables:






SELECT * FROM Publisher
publisherName publisherId
Bloomsbury 1
NCERT 2
CBSE 3
Penguin 4
Simon 5
Query executed successfully at 21:26:02


SELECT * FROM Bookstore
bookstoreName address
ASU Bookstore ASU
Arizona Bookstore AZ
Chandler Bookstore Chandler
Gilbert Bookstore Gilbert
Mesa Bookstore Mesa
Polytechnic Bookstore POLYTECHNIC, ASU
Tempe Bookstore Tempe
Query executed successfully at 21:34:43

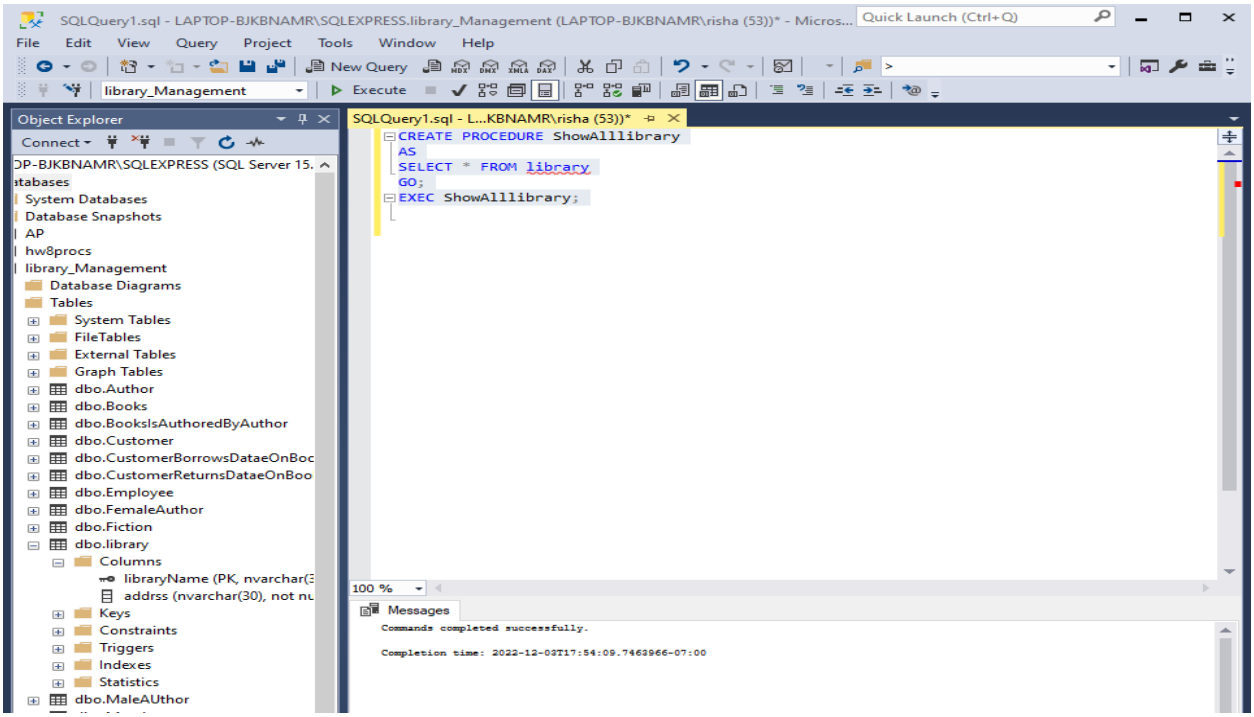
Section 5 – Additional constraints

Stored Procedures:

CODE:

```
CREATE PROCEDURE ShowAlllibrary

AS

SELECT * FROM library

GO;

EXEC ShowAlllibrary;
```

SCREENSHOT:



2.

CODE:

CREATE PROCEDURE ListOfEmployees

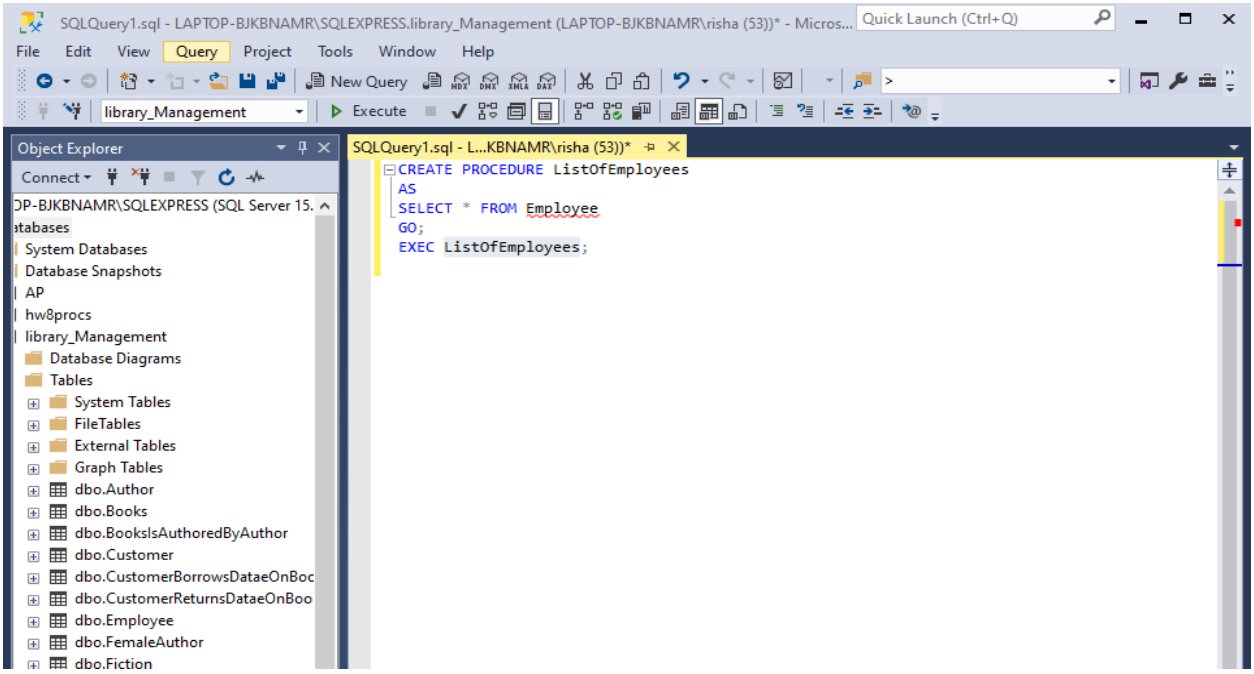
AS

SELECT * FROM Employee

GO;

EXEC ListOfEmployees;

SCREENSHOT:



3.

CODE:
CREATE PROCEDURE ShowAllPublishers

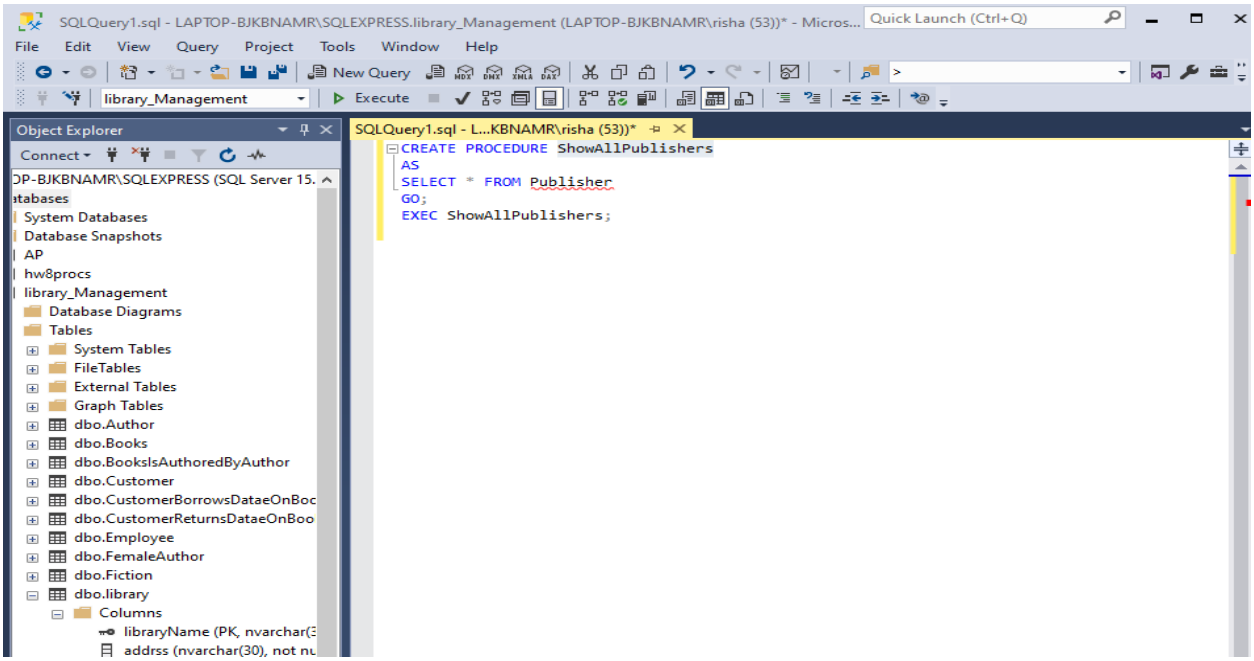
AS

SELECT * FROM Publisher

GO;

EXEC ShowAllPublishers;

SCREENSHOT:



TRIGGERS:

1. **CODE:**

   CREATE TRIGGER DisplayDeletedLibrary

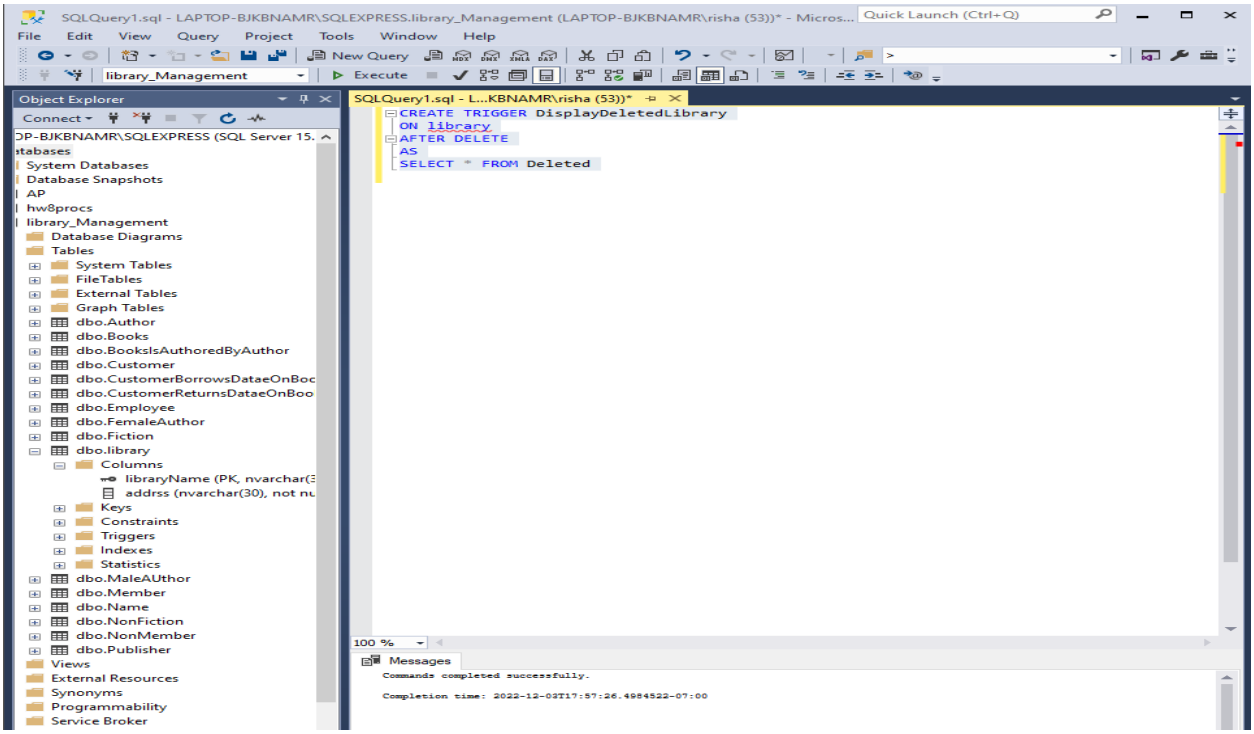
   ON library

   AFTER DELETE

   AS

   SELECT * FROM Deleted

   SCREENSHOT:



2. **CODE:**

CREATE TRIGGER DisplayUpdatedTable

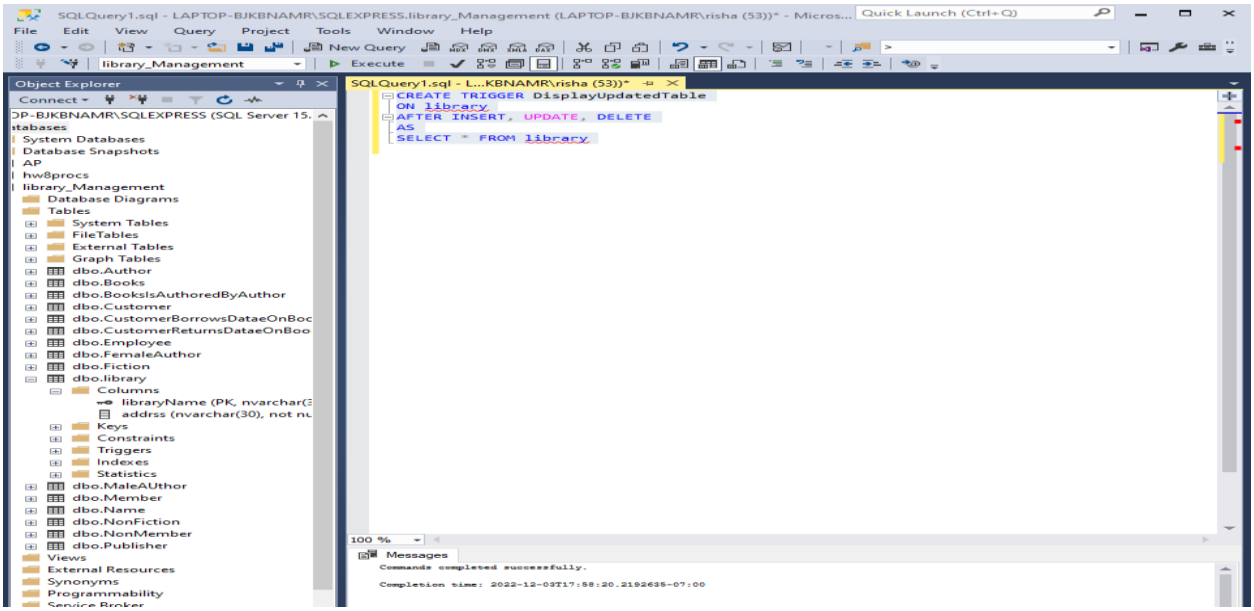
ON library

AFTER INSERT, UPDATE, DELETE

AS

SELECT * FROM library

SCREENSHOT:

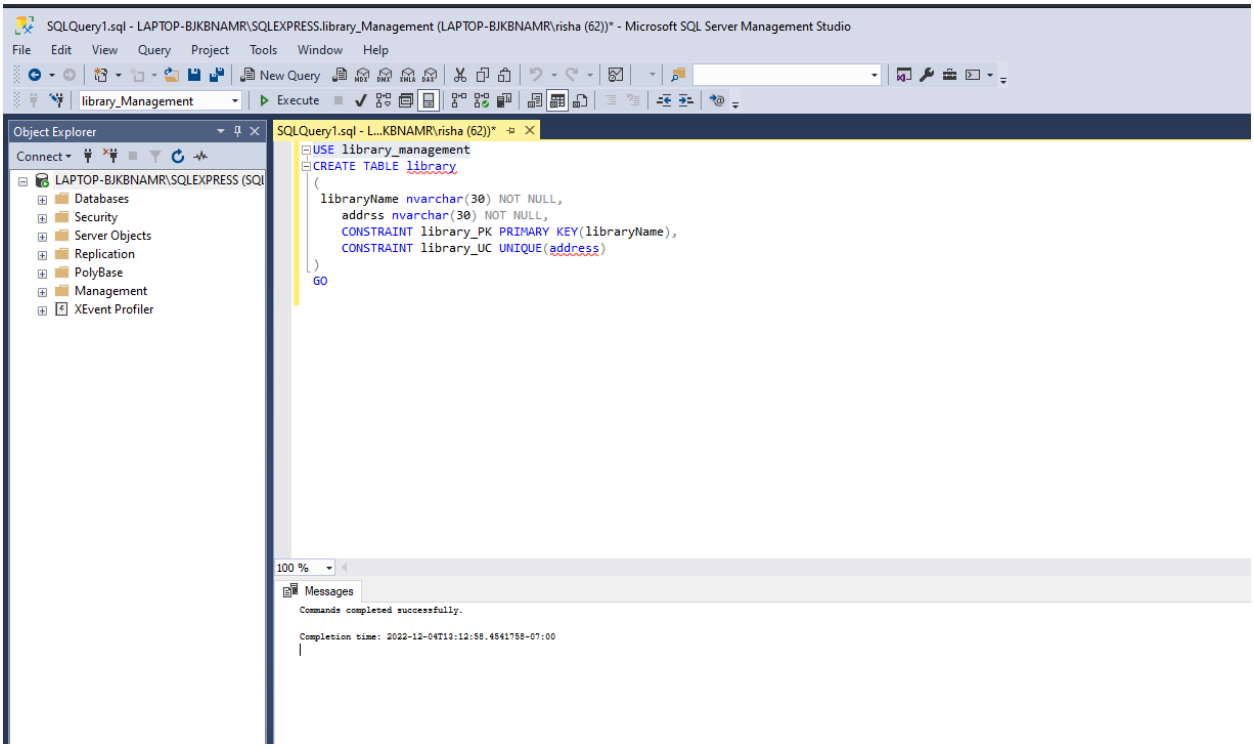


Section 5.1:

1. **Create Table Query:**

```
USE library_management

CREATE TABLE library

(

 libraryName nvarchar(30) NOT NULL,

        addrss nvarchar(30) NOT NULL,

        CONSTRAINT library_PK PRIMARY KEY(libraryName),

        CONSTRAINT library_UC UNIQUE(address)

)

GO
```

SCREENSHOT:



2. **Alter Table Query:**

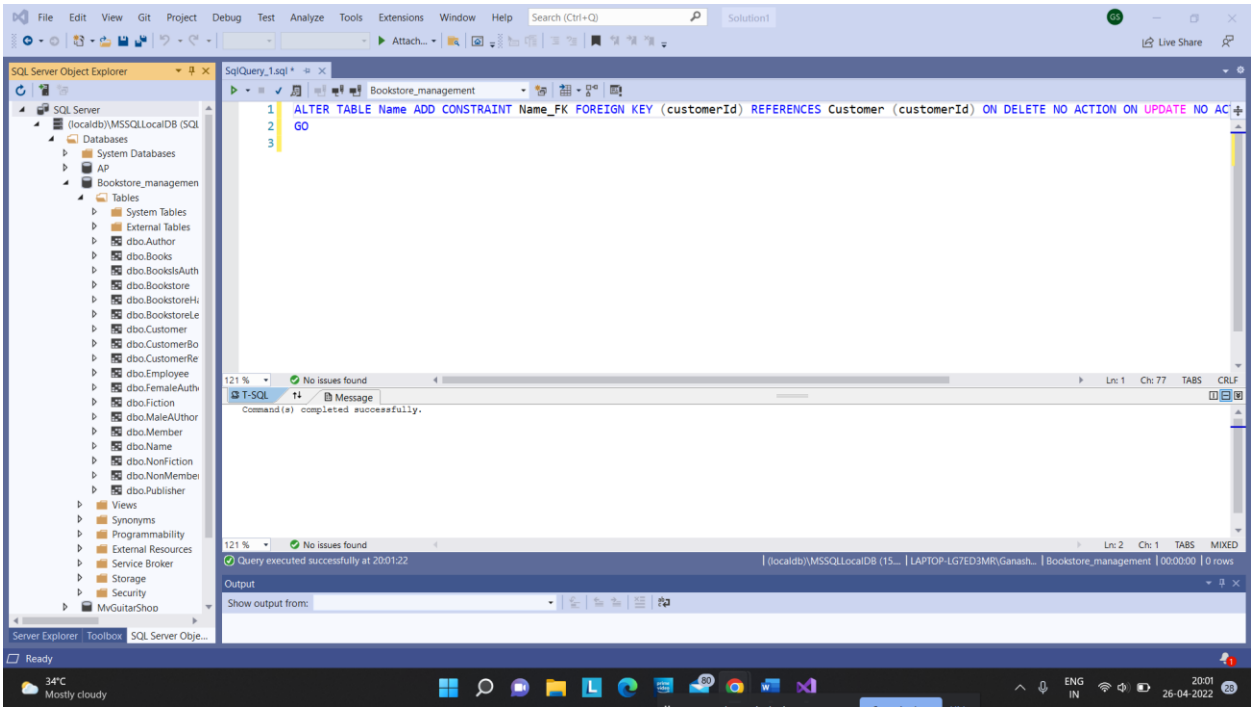
   ALTER TABLE Name ADD CONSTRAINT Name_FK FOREIGN KEY

   (customerId) REFERENCES Customer (customerId) ON DELETE NO ACTION

   ON UPDATE NO ACTION

   GO

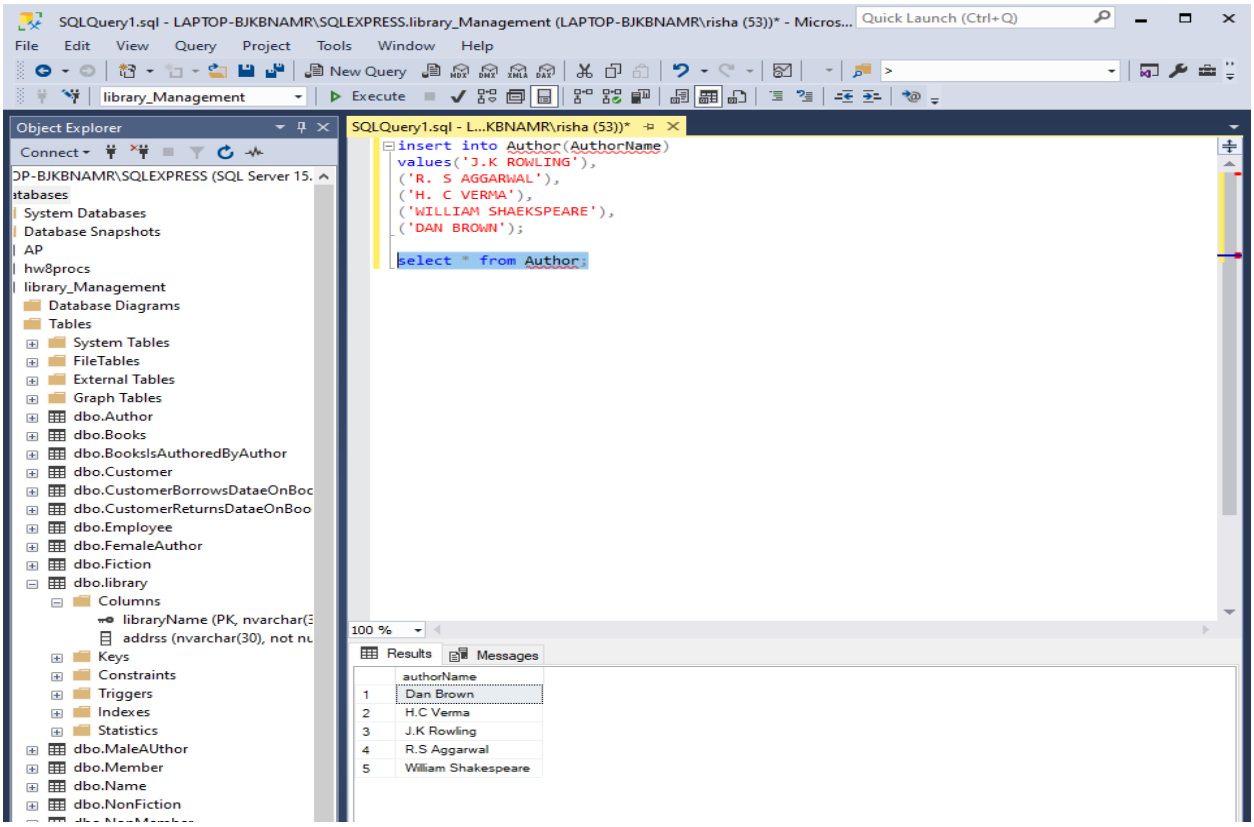
   SCREENSHOT:



3. **Insert value Query:**

```
Insert into Author (authorName)

values ('J.K Rowling'),

('R.S Aggarwal'),

('H.C Verma'),

('William Shakespeare'),

('Dan Brown');
```

SCREENSHOT:



4. **Trigger Query:**

CREATE TRIGGER DisplayDeletedStore

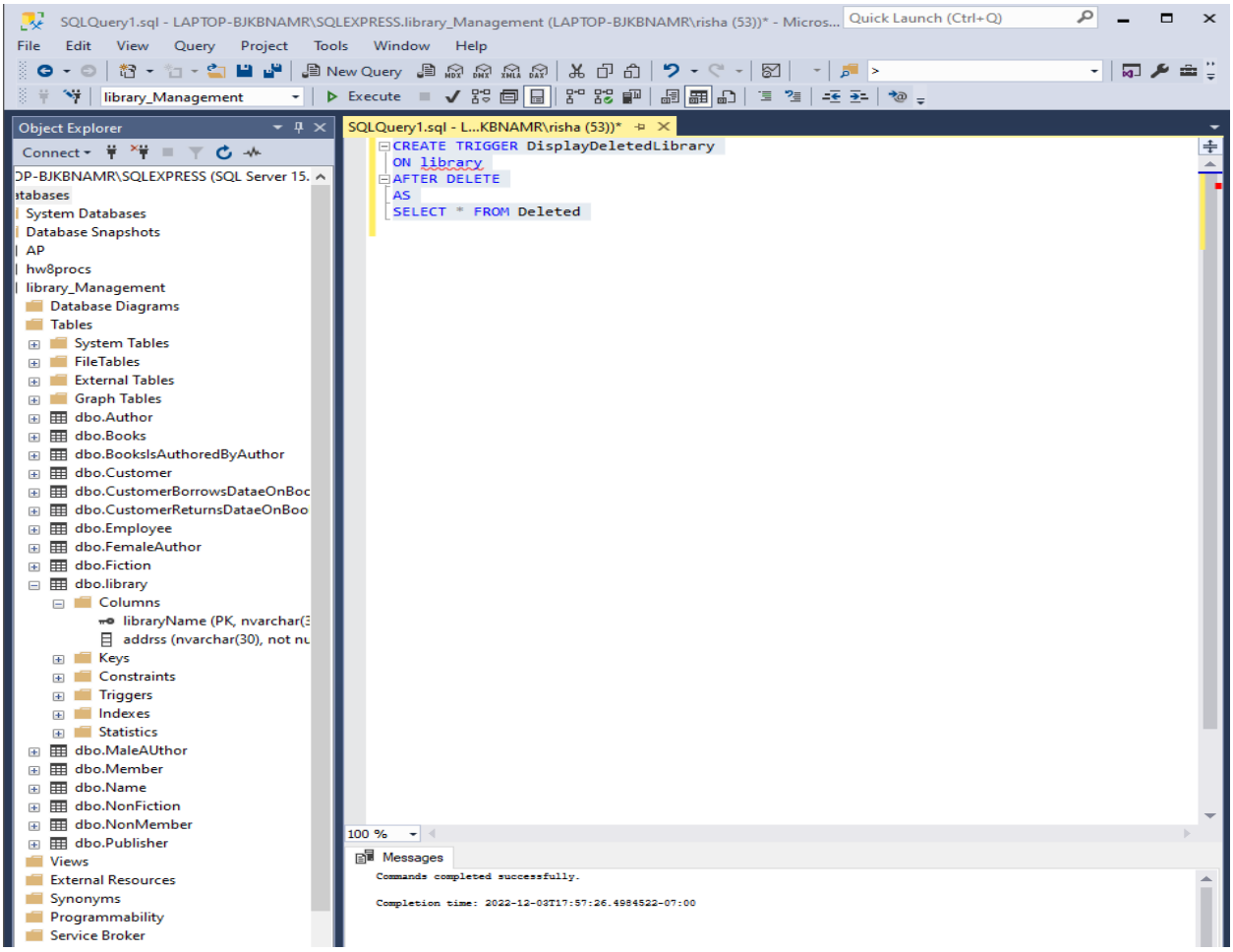
ON library

AFTER DELETE

AS

SELECT * FROM Deleted

Screenshot:



5. **Stored Procedure:**

CREATE PROCEDURE ShowAlllibrary

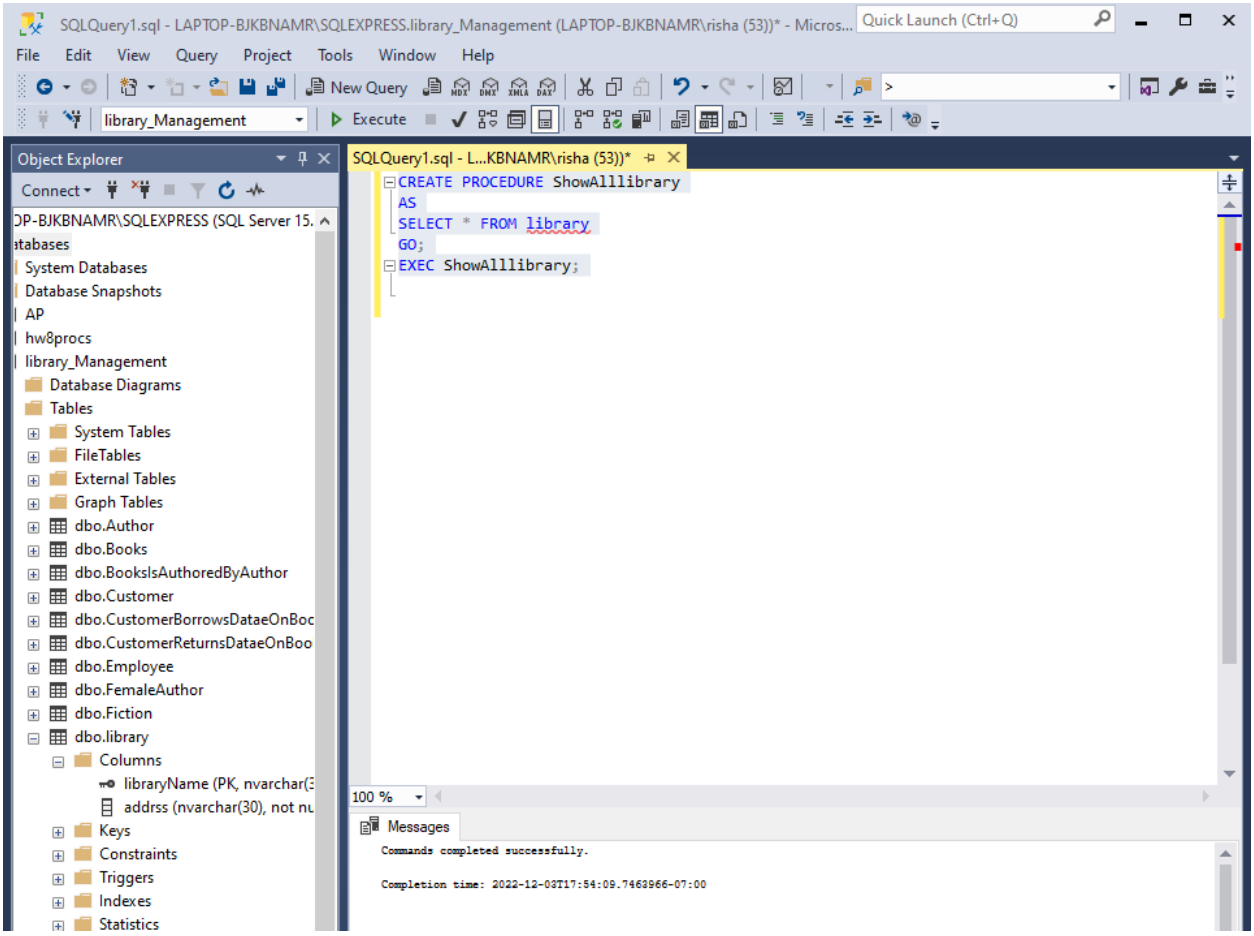
AS

SELECT * FROM library

GO;

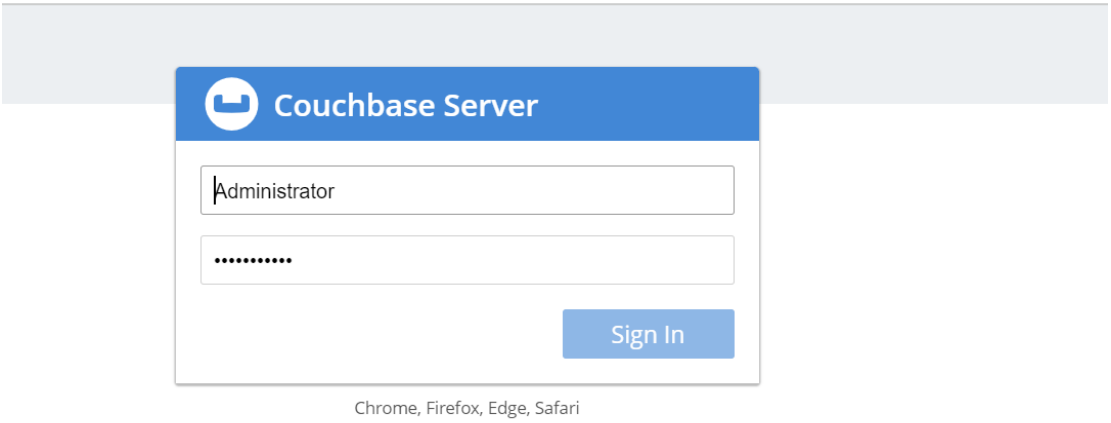
EXEC ShowAlllibrary;

SCREENSHOT:


SQLQuery1.sql - LAPTOP-BJKBNAMR\SQLEXPRESS.library_Management (LAPTOP-BJKBNAMR\risha (53))* - Micros...
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
library_Management
Execute
Object Explorer
Connect
OP-BJKBNAMR\SQLEXPRESS (SQL Server 15.
tabases
System Databases
Database Snapshots
AP
hw8procs
library_Management
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Author
dbo.Books
dbo.BooksIsAuthoredByAuthor
dbo.Customer
dbo.CustomerBorrowsDataeOnBoc
dbo.CustomerReturnsDataeOnBoo
dbo.Employee
dbo.FemaleAuthor
dbo.Fiction
dbo.library
Columns
libraryName (PK, nvarchar(3
addrss (nvarchar(30), not nu
Keys
Constraints
Triggers
Indexes
Statistics
SQLQuery1.sql - L...KBNAMR\risha (53))*
CREATE PROCEDURE ShowAlllibrary
AS
SELECT * FROM library
GO;
EXEC ShowAlllibrary;
100 %
Messages
Commands completed successfully.
Completion time: 2022-12-03T17:54:09.7463966-07:00

SECTION 6: COUCHBASE(NOSQL) Configuration

We have created Couchbase enterprise account and it is ready to use



Wide-column stores, document databases, graph databases, key-value pairs, and non-relational databases (or NoSQL databases) are a few examples. You have more freedom when handling "unstructured data" since NoSQL databases don't need a schema.

NoSQL databases are made to manage more intricate, unstructured data, which makes up a growing portion of today's data (including texts, social media postings, images, videos, and email). Keyspaces, which are enormous collections of free-form text, are used by N1QL to organize information. Data reshaping is provided by N1QL by incorporating statement-attributes into the intended result-object structure.

A collection of databases called a NoSQL database system can store structured, semi-structured, unstructured, and polymorphic data.

There are two main choices for access:

1. REST APIs: To submit a request to an endpoint with specific features is referred to in this sentence.

2. CRUD (create, read, update, delete) in vendor-specific language: You'll observe that Mongo DB has a distinct method for carrying out queries if you use it.

The Couchbase constructs are as follows:

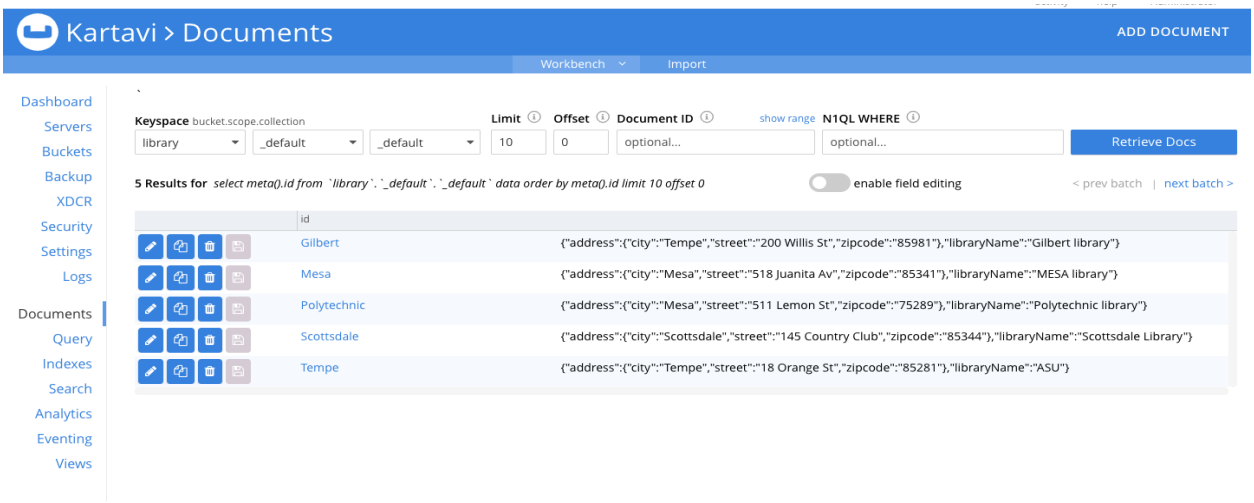
- Cluster
- Bucket
- Scope
- Collection
- Document (JSON)

Due to the ability to extend capacity horizontally across low-cost, commodity servers, scaling a NoSQL database is much less expensive than scaling a relational database.

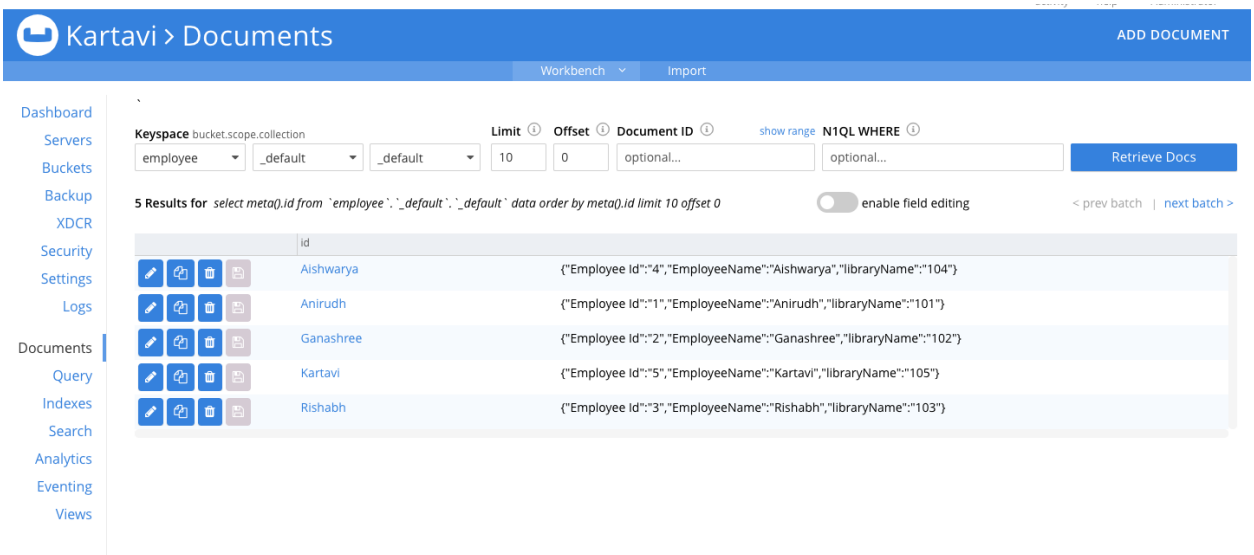
Section 6.1 : ORM diagram from a NoSQL perspective

The entities that we are extending through nesting are the employee table and the bookshop (address field). To connect the two entities, utilize the field libraryName.

Documents of libraries

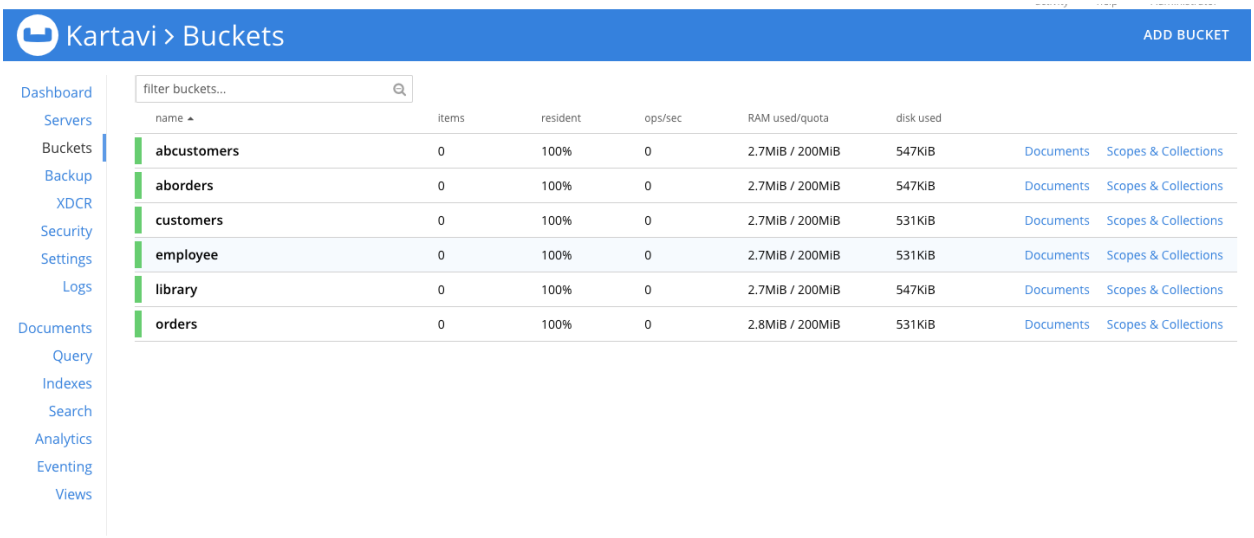


Document of Employees

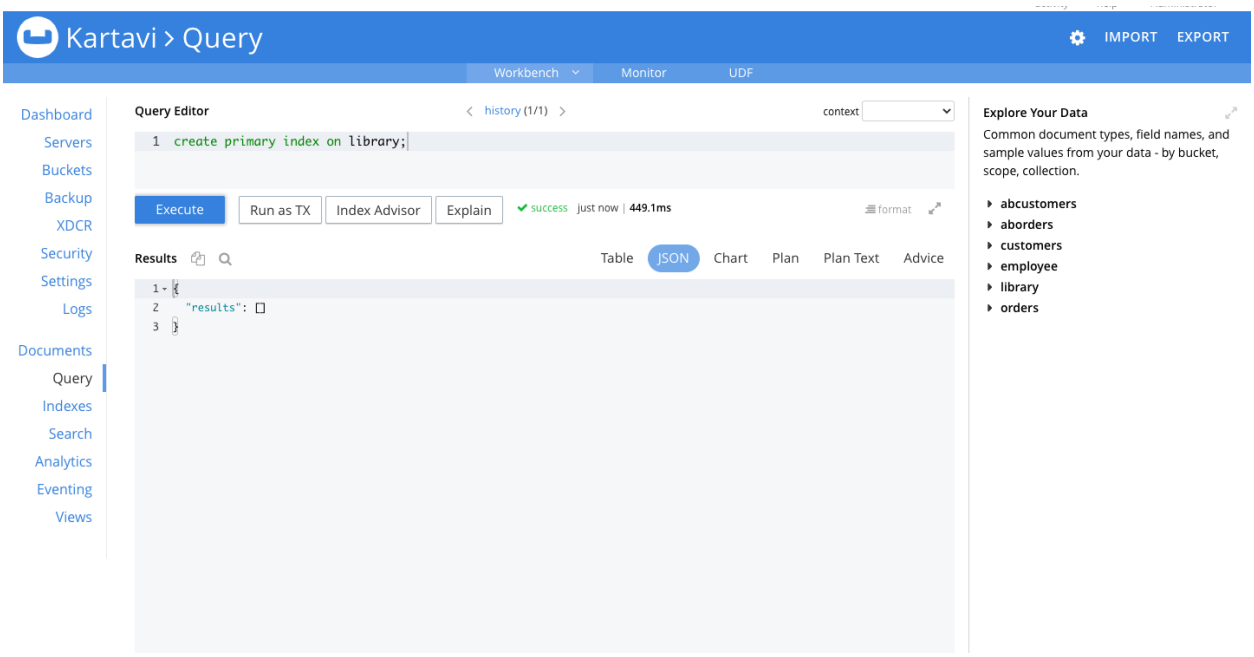


Section 6.2

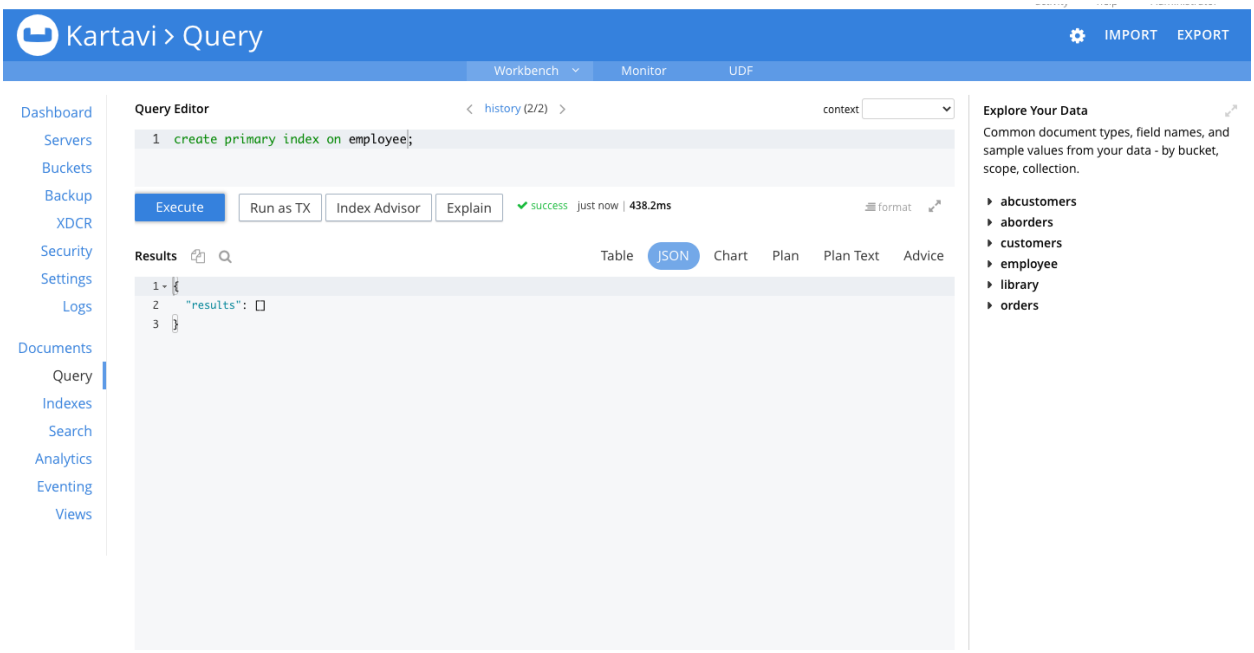
Created Buckets



Created Primary Index on library

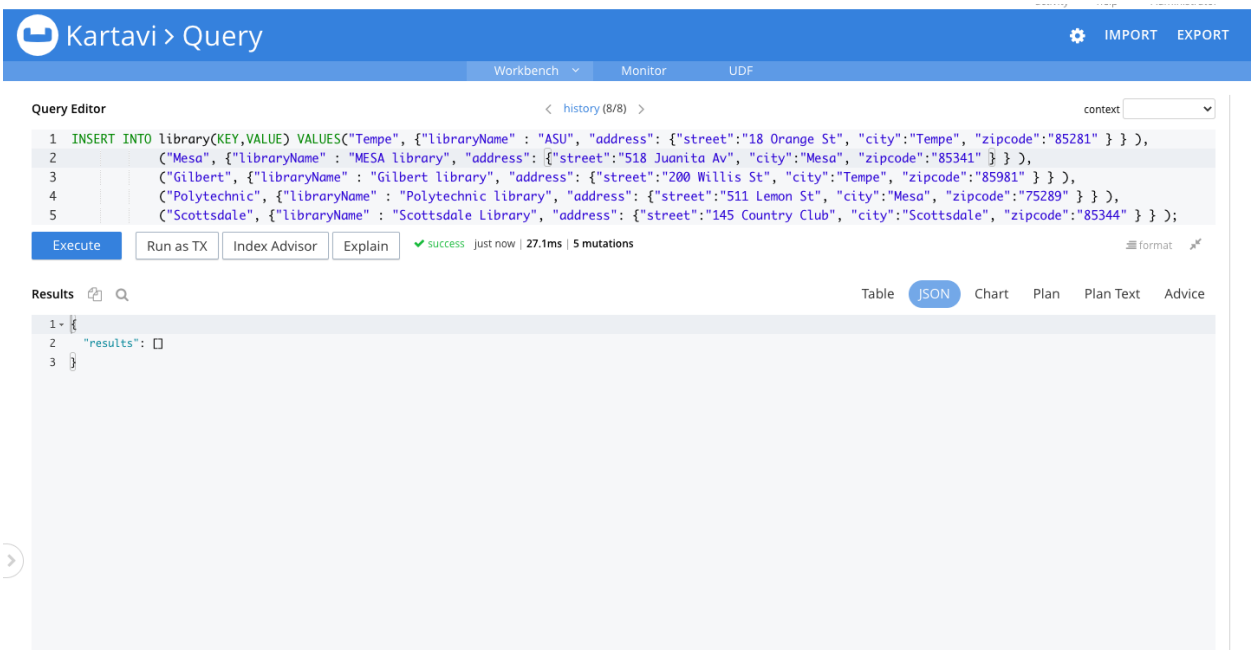


Created primary index on Employee

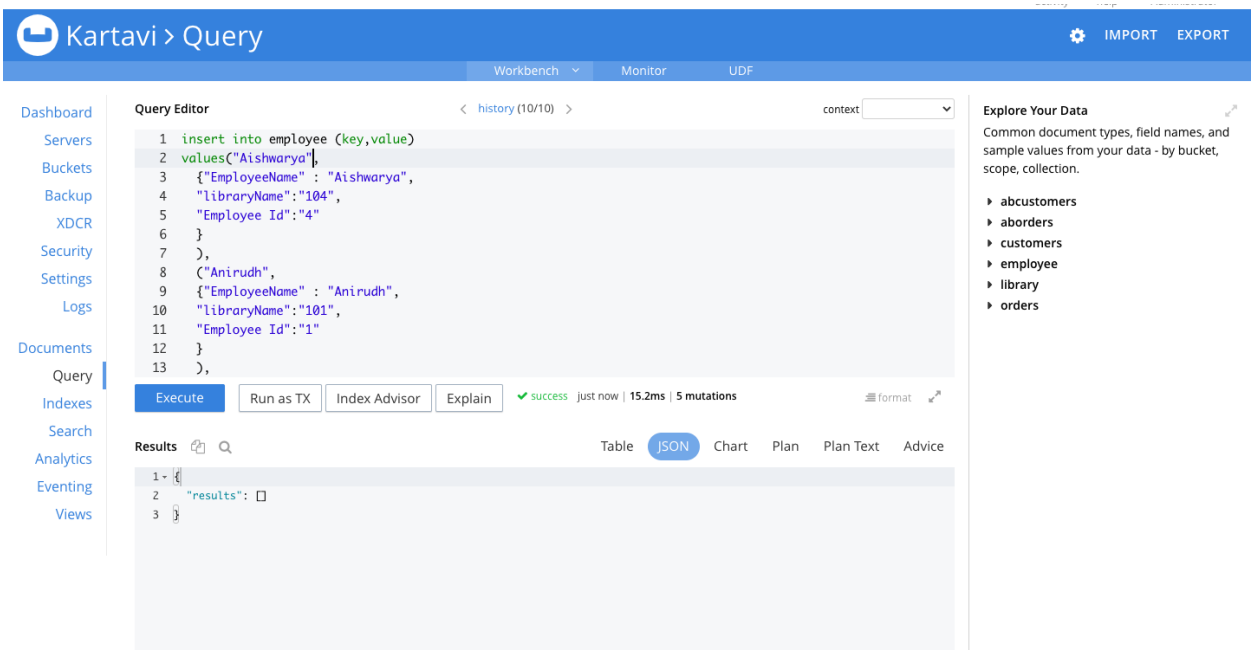


Section 6.3

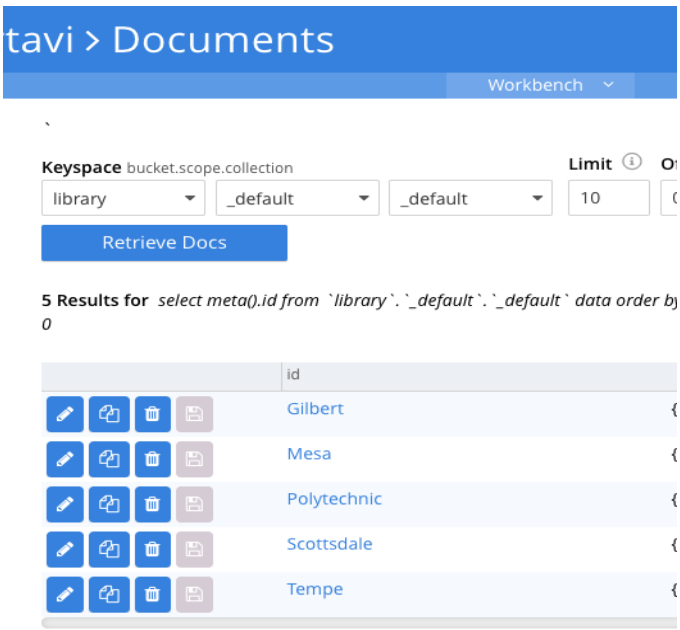
Inserting values into library Table

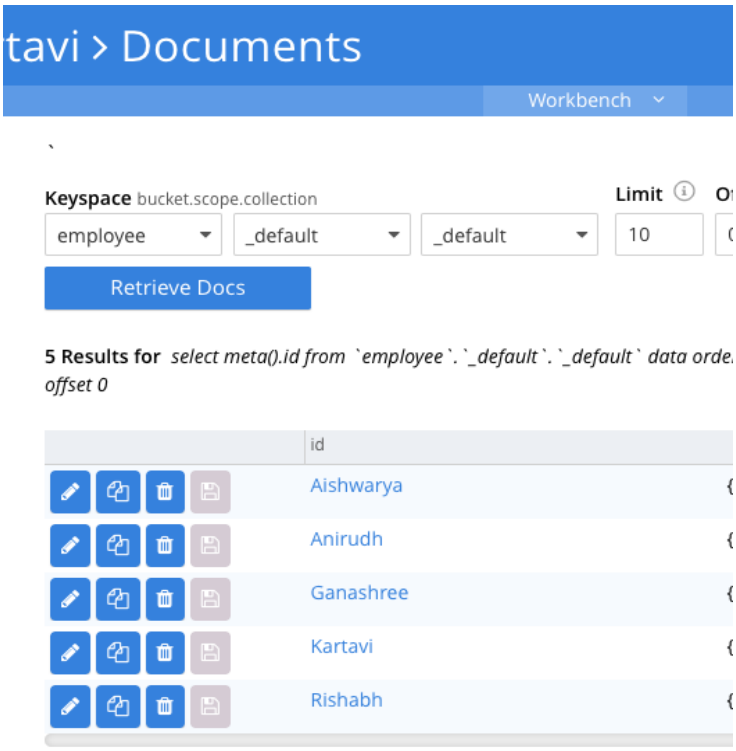


Inserting Values in Employee table



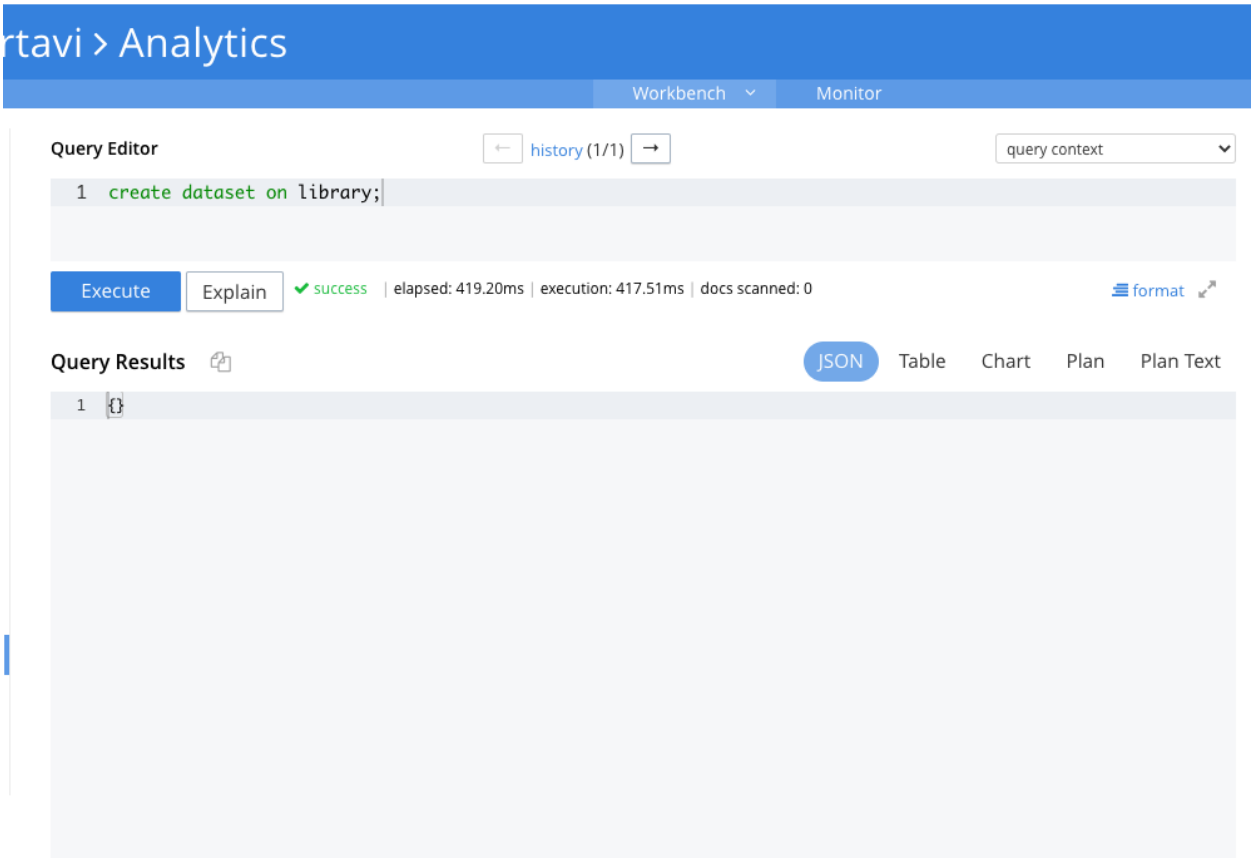
Json to Represent Data



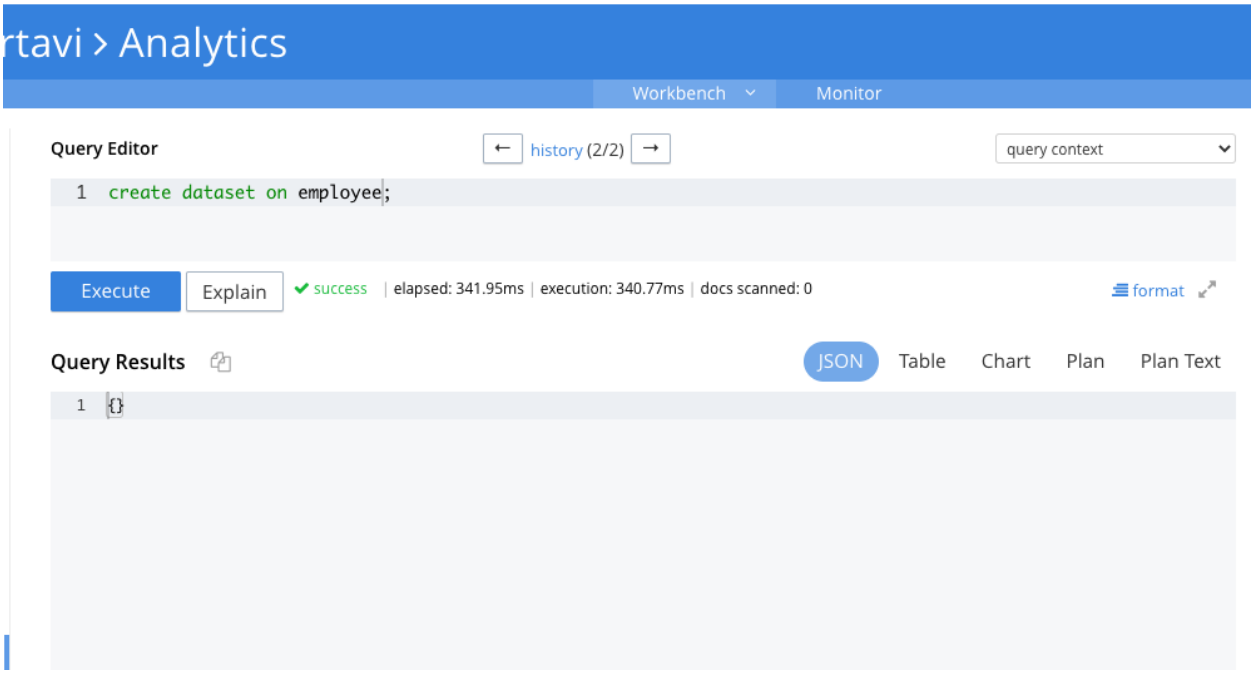

tavi > Documents
Workbench
Keyspace bucket.scope.collection
employee
_default
_default
10
Limit
Retrieve Docs
5 Results for select meta().id from `employee`.`_default`.`_default` data orde
offset 0
id
Aishwarya
Anirudh
Ganashree
Kartavi
Rishabh

Section 6.4

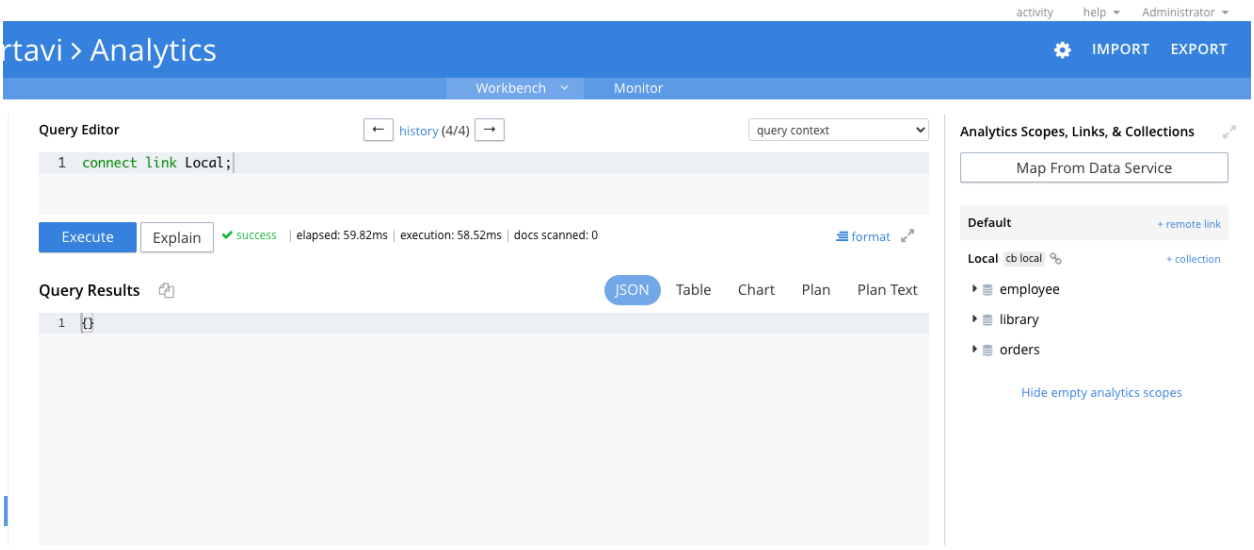
Creating dataset on library



Create dataset on employee



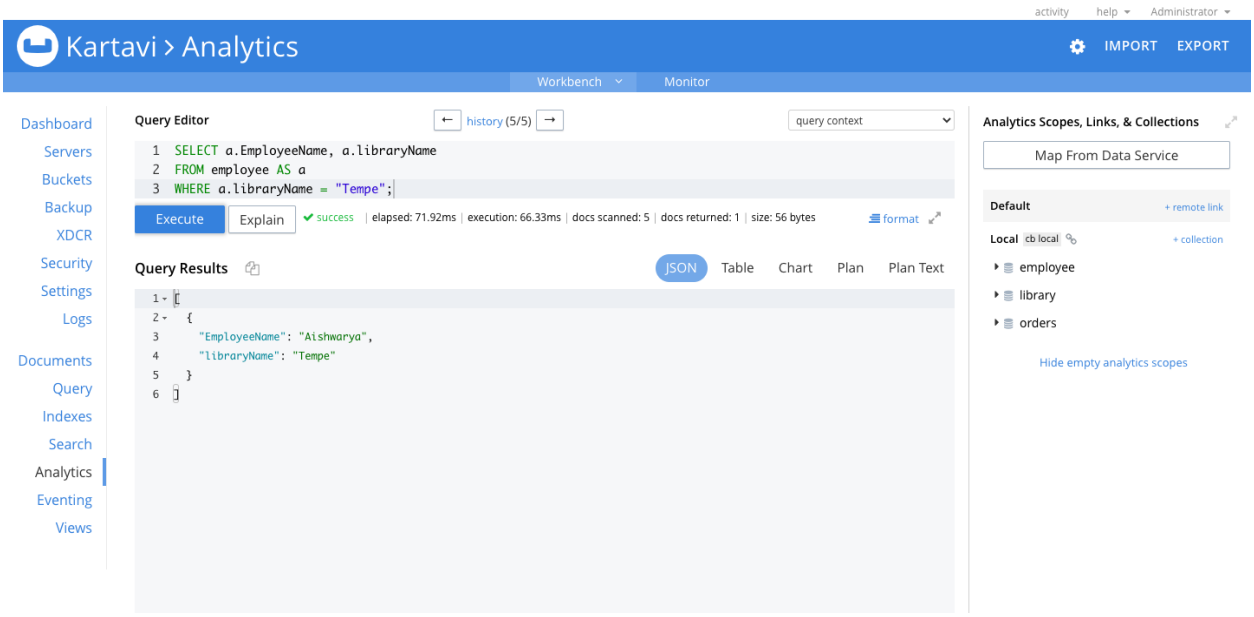
Connected the local link



Running queries using analytics service

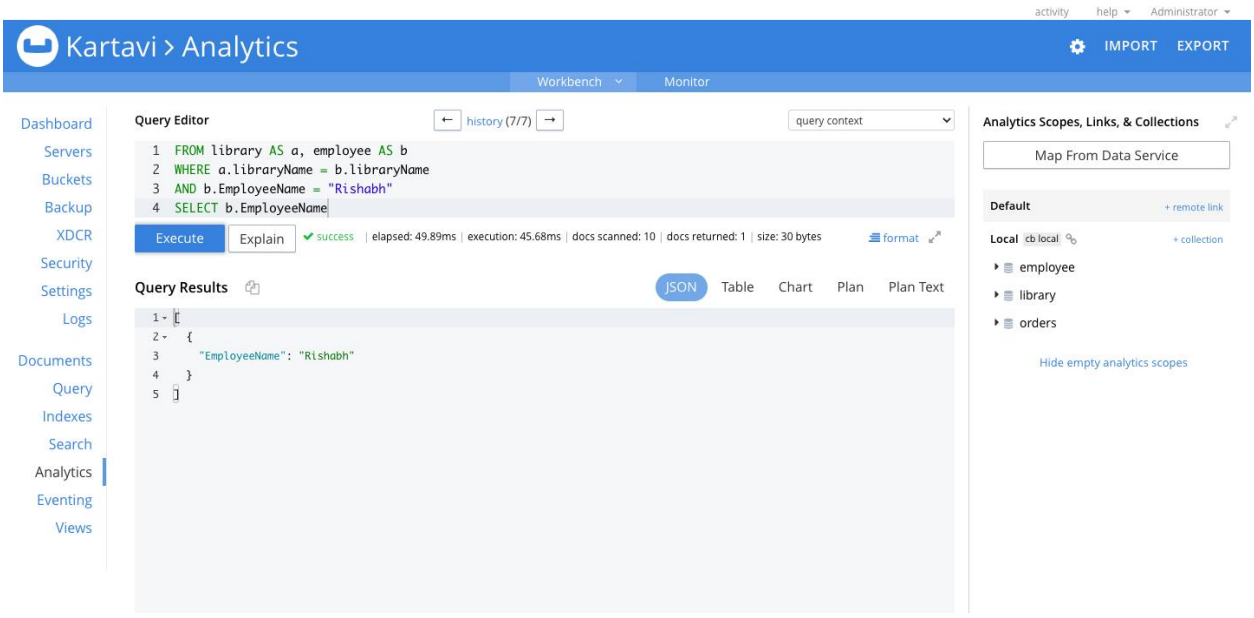
Code:

```
SELECT a.EmployeeName, a.libraryName

FROM employee AS a

WHERE a.libraryName = "Tempe";
```



Code:

```
FROM library AS a, employee AS b

WHERE a.libraryName = b.libraryName

AND b.EmployeeName = "Gowthami"

SELECT b.EmployeeName
```



Code:

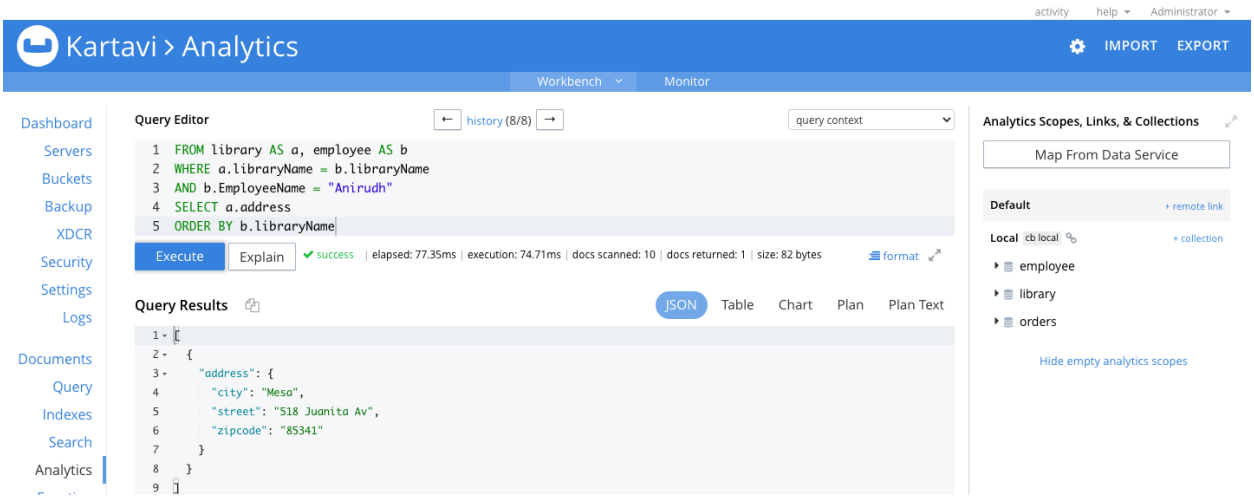
FROM library AS a, employee AS b

WHERE a.libraryName = b.libraryName

AND b.EmployeeName = "Anirudh"

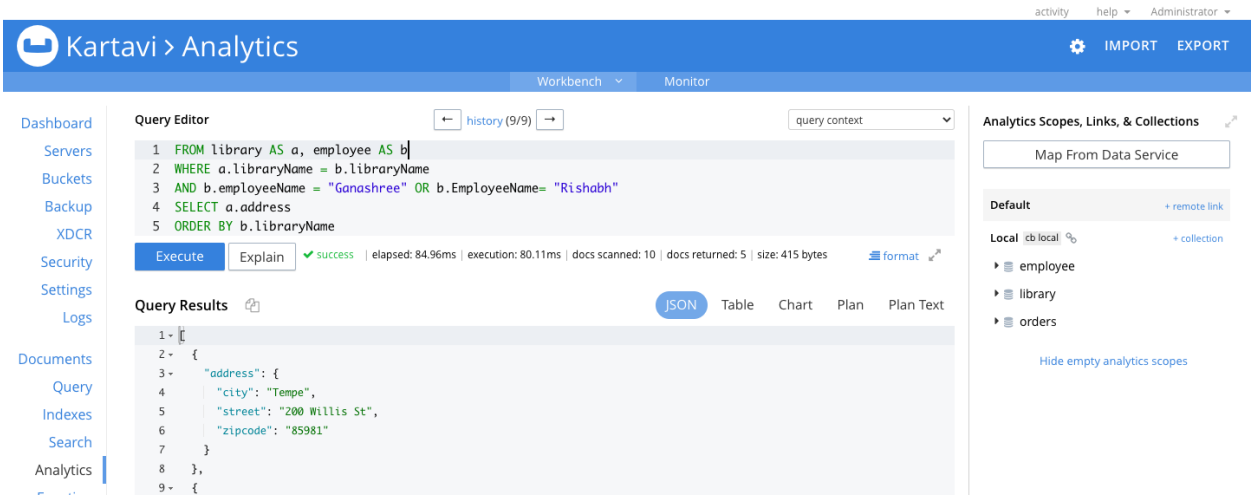
SELECT a.address

ORDER BY b.libraryName



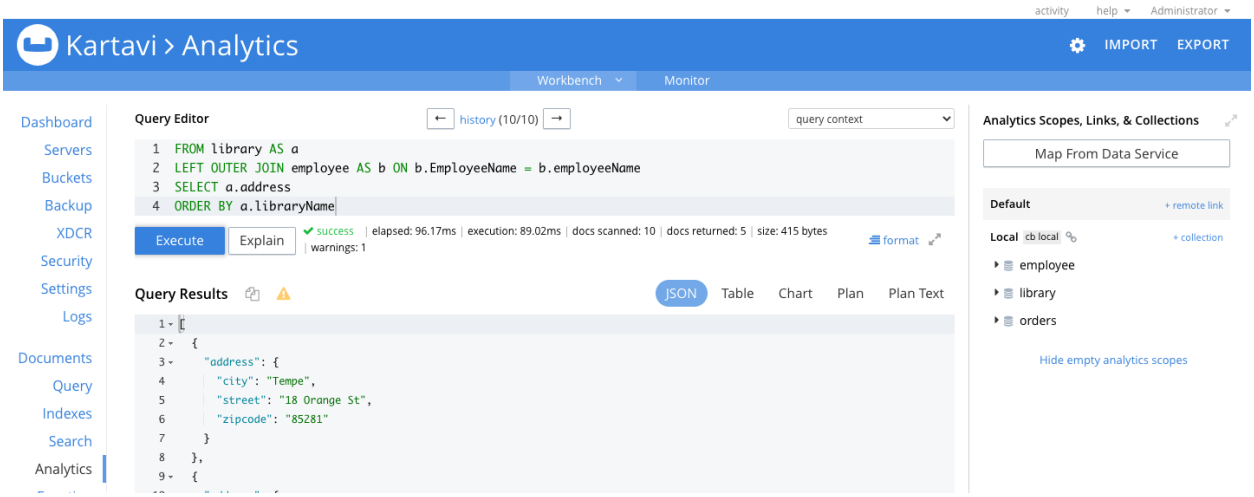
Code:

```
FROM library AS a, employee AS b

WHERE a.libraryName = b.libraryName

AND b.employeeName = "Ganashree" OR b.EmployeeName= "Rishabh"

SELECT a.address

ORDER BY b.libraryName
```



Code:

```
FROM library AS a

LEFT OUTER JOIN employee AS b ON b.EmployeeName = b.employeeName

SELECT a.address

ORDER BY a.libraryName
```


activity
help
Administrator
Kartavi > Analytics
IMPORT
EXPORT
Workbench
Monitor
Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Query Editor
history (10/10)
query context
1 FROM library AS a
2 LEFT OUTER JOIN employee AS b ON b.EmployeeName = b.employeeName
3 SELECT a.address
4 ORDER BY a.libraryName
Execute
Explain
success | elapsed: 96.17ms | execution: 89.02ms | docs scanned: 10 | docs returned: 5 | size: 415 bytes | warnings: 1
format
Query Results
JSON
Table
Chart
Plan
Plan Text
[
{
"address": {
"city": "Tempe",
"street": "18 Orange St",
"zipcode": "85281"
}
},
{
Analytics Scopes, Links, & Collections
Map From Data Service
Default
+ remote link
Local
cb local
+ collection
employee
library
orders
Hide empty analytics scopes

Section 7

Summary:

For this project, an Object-Role modeling diagram from which the relational model view was formed was created using the Conceptual Schema Design Procedure. Then, we wrote SQL code and implemented it using SQL Server Management Studio on MS SQL, building a database master, schema Schema1, and tables bookshop, employee, Author, Books, Publisher, and customer, and adding data into them. In order to put particular protocols on the database, we then applied the required constraints. Then, we added triggers, which cause a planned action to take place in response to a given occurrence. Additionally, we created stored procedures to improve the SQL scripts' reuse.

Then, we built a Couchbase NoSQL database and two Buckets: employee and bookshop. After indexing and putting data on the Query service, we constructed datasets on both buckets and connected Query service data with Analytics service data. To execute SQL++ queries, including creating a JOIN over nested documents, we used the Analytics API. By keeping track of patron information and maintaining books along with author and publisher data in an employee-run library, we were able to successfully address the difficulties raised in the first proposal.

Conclusion

We built SQL and NoSQL databases for this project with the help of MSSQL and Couchbase, respectively. We were able to determine the differences between NoSQL and SQL as a result.

As we taught, the table-based data structure of SQL databases has a strict, predefined schema. However, NoSQL databases do not need a schema, which gives you more freedom when working with "unstructured data." When a SQL query is executed, a collection of rows is created, with the same columns present in all rows and one or more columns in each row. N1QL organizes data into key-value pairs, in contrast to the way that large collections of free-form texts are organized by N1QL. Data reshaping is provided by N1QL by incorporating statement-attributes into the intended result-object structure. CRUD operations are used by several relational database engines, such as Microsoft SQL Server, PostgreSQL, and MySQL. Since queries are typically made in CRUD syntax when it comes to access, raw SQL is usually employed. REST APIs and CRUD activities are used to get the data in NoSQL, though. RDBMS scales vertically, but NoSQL scales horizontally.

This might be developed into a GUI-based program that offers users portals based on their access privileges.

References:

1. **https://docs.couchbase.com/server/current/introduction/intro.html**
2. **https://www.couchbase.com/sql-plus-plus-for-sql-users**