

TASK 1

LIBRARY MANAGEMENT SYSTEM

INTRODUCTION

A library is a place where a huge collection of books and resources are available which can be accessible by the users. It enhances the dissemination of knowledge and spiritual civilization among the members. The tons of items (eg. books and other resources) and research works are captivating the members to improvise their knowledge in all perspectives. It guides the members to promote their views differently. This knowledge optimizes the members to achieve better result in the development of their various personal skills and awareness. Improvisation in technology causes the demand for developing a way to enhance the traditional library set up to digital one. Numerous tedious processes reduce the efficiency of the library. For example, it always needs manual support to do any activities in the traditional library. The count and details of library items are scribbled in the paper for reference. Each data is fetched in the notebook for future citations. To examine any data then they have to refer the notebooks. At the same time while distributing the items of the library to members they have to enter into the notebook where they need to represent the item id, distribution and return date, and member id. The librarians/staff have to assign a tag for each item and provide an id for it. They have to align and arrange the items on the shelves and mark it. Missing or theft of the items in the library builds a serious issue and confusion to the librarians. While collecting the book from the members they have to verify the penalties of the books. Therefore it causes a monotonous among the staff. Consequently, it builds an uninteresting among the members due to the slow progress of the staff.

To evoke the library into the technological era, we are presented a system called the Library Management system (LMS). It is an automatic system that reduces the work burden of the staff/librarians through a single click. It will manage, organize and oriented the library task. The LMS supports the librarian to add/view/delete/update details from the library stock. Here we integrate all the library data into the SQL server. Preliminarily the librarian has to add members and library items details into the database. After that he/she can

view/delete/update those details through the Library Management system. On account of this, the user can access the library at any time. The librarians can assist the data without any confusion. Each data are retrieved from the database. if he/she access any user details then it shows username, id, item details, and penalty details. They no longer need to write it on paper for any references. By editing the data they can change the parameter in it. In spite of working on the manual, the librarian can find it easy to handle the automatic system. It has more additional features such as librarian can maintain library records, member's history of penalties and issues. It always tracks the count of the items in the library and issued item details. This causes a flexible service for librarians and members of the library. It is a user-friendly interface, so basic computer knowledge is enough to access the LMS. The system is customizable and user-configurable one which causes it to use in different organizations. We represent the LMS with Admin module.

As aforementioned the data are stored and secured in the database. The related data are stored together and maintained properly. It allows the user to create their database as per the requirement. The database gets queried by the programs which provide an interface between the databases. The database management system (DBMS) receives the command from the administrator and based on the instruction it queries the data in the database. This instruction may load, retrieve or modify the existing database. It is better to assign a DBMS as a centralized one which helps multiple users to access the database in a controlled manner at a different location. Based on the scheme of DBMS, the system can assign a view mode for each user like some people can see only some data and authorized one

can see all the data existing in the database. It offers both logical and physical data independence.

1. DATABASE DESIGN

In this document, I will be designing a database based on a given set of requirements. This system will be responsible for managing the various items which a library has in stock as well as managing the various items which a member has borrowed and returned. As part of this software development, it is important to be guided by the set of requirements that is expected to be met. The list below are the requirements that this database is required to meet.

- (a) Members must provide their full name, address, date of birth and they must create a username and password to allow them to sign into the member portal, they can also provide an email address and telephone number.**
- (b) Fines for borrowed items that are overdue and the library has to keep track of the total overdue fines owed by a member, how much they have repaid and the outstanding balance.**
- (c) The library wants to retain the information of members who leaves so they can continue marketing to them, but they should keep a record of the date the membership ended.**
- (d) Each repayment needs to be recorded, along with the date / time of the repayment, the amount repaid and the repayment method (cash or card).**

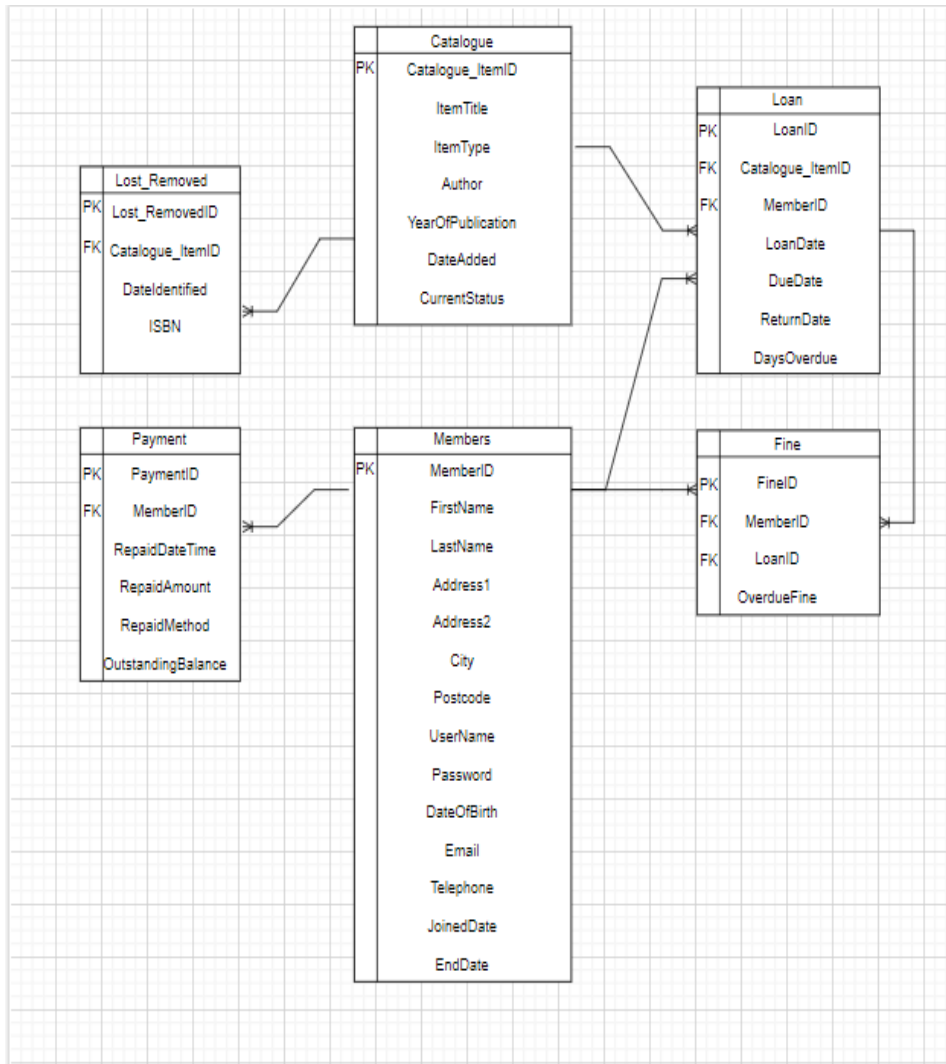
(e)The Items of the library catalogue of items must be recorded as item title, item type (which is classified as either a Book, Journal, DVD or Other Media), author, year of publication, date the item was added to the collection and current status (which is either On Loan, Overdue, Available or Lost/Removed).

(f)When an item is lost or removed from the library, the date this was identified will be recorded. If it is a book, the ISBN will be recorded.

(g)Each loan should specify the member, the item, the date the item was taken out, the date the item is due back and the date the item was actually returned. If the item is overdue, an overdue fee needs to be calculated at a rate of 10p per day.

With respect to the specifications highlighted above, I will start by preparing an entity relationship diagram which is a diagram that contains a set of tables, columns and how they are related.

The ER Diagram



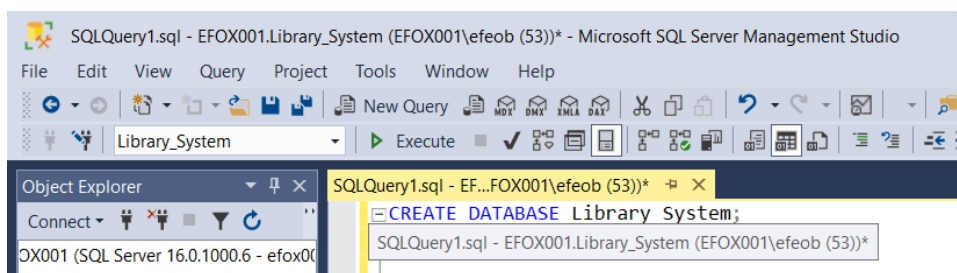
In this database design, I have provided for six tables which is based on the stated requirements and they are Catalogue, Members, Lost_Removed, Payment, Loan and Fine. As you can see from the ER diagram, each of the tables have been assigned several attributes with the aim of establishing a relationship between each of the table.

SQL DATABASE

SQL is also known as Standard Query Language which is used as a medium for communicating with the database. SQL statements are utilized to execute the queries against the database and retrieve data from the database. We

can create a new database, table, stored procedure and update, delete and add items to the table. We also can view the data and set permission to the view, procedure, and table.

I will now create a database for the Library Management System which will be used to record data on the items within the library catalogue, members of the library, and other activities within the library to aid the library admin in managing the operations of the library. Using the simple code “CREATE DATABASE” I shall create the database called Library_System.



This database will definitely be associated with several tables which has been represented by the six different entities highlighted in the ER Diagram above. Each entity in the Entity Relational Diagram above represents a table in the database and the attributes associated with each of these entities represents the columns within the tables to be created. As this is a relational database management system (RDMS), each of these entities has been assigned a primary key (PK), and one or more foreign keys (FK) which defines the relationship among the entities in the ER Diagram and when implemented in creating the tables, it will therefore define the relationship/connection among the tables of the database.

Another thing showing the relationship among the entities in the ER Diagram is the Entity Relation Lines indicates the nature of the relationship between two entities. For example, the line connecting Catalogue and Loan indicates a one to many relationship which is so because the primary key in the Catalogue can be connected to more than one Loans but the primary key in Loan can only be connected to one item in the Catalogue. This same

relationship is repeated between Catalogue and other entities connected to Catalogue, it is also repeated between Member and the entities connected to Member, and finally it exists between Loan and Fine entities.

CREATE TABLES

After concluding with the ER Diagram and creating the database, the next step would be creating the various tables associated with the database that has been created. In creating the tables using the sql statement “CREATE TABLE”, the various columns associated with each table will be indicated in the statement along with the data types and the respective constraints.

Using the Microsoft SQL Management Studio, it is important to ensure that the database to be worked on is set at Library_System which is the database for this project.

Below is the SQL statement to create the first table.

-- Create Catalogue table

```
USE Library_System

CREATE TABLE Catalogue (
    CatalogueID int NOT NULL Primary Key,
    ItemTitle nvarchar(30) NOT NULL,
    ItemType nvarchar(15) NOT NULL,
    Author nvarchar(30) NOT NULL,
    YearOfPublication Int NOT NULL,
    DateAdded Date NOT NULL,
    CurrentStatus nvarchar(15) NOT NULL);
```

The second table would be recording all items that has been borrowed from the library, the member that has borrowed an item, the date it was borrowed, the due date, the date it was actually returned and the days overdue.

--Create Loan table

```

CREATE TABLE Loan (
  LoanID int NOT NULL Primary Key,
  CatalogueID int NOT NULL Foreign Key
  (CatalogueID) References Catalogue (CatalogueID),
  MemberID int NOT NULL Foreign Key
  (MemberID) References Members (MemberID),
  LoanDate Date NOT NULL,
  DueDate Date NOT NULL,
  ReturnDate Date NULL,
  DaysOverdue int NULL);

```

The next table holds information on the members of the library which are collected and stored as the member joins the library.

--Create Members table

```

CREATE TABLE Members (
  MemberID int NOT NULL Primary Key,
  FirstName nvarchar(20) NOT NULL,
  LastName nvarchar(20) NOT NULL,
  Address1 nvarchar(50) NOT NULL,
  Address2 nvarchar(50) NULL,
  City nvarchar(50) NULL,
  Postcode nvarchar(10) NOT NULL,
  DateOfBirth Date NOT NULL,
  UserName nvarchar(30) NOT NULL,
  Password nvarchar(20) NOT NULL,
  Email nvarchar(100) UNIQUE NULL
  CHECK(Email LIKE '%_@_%._%'),
  Telephone int NULL,
  DateJoined Date NOT NULL,
  EndDate Date NULL);

```

Another table in this database is the table that holds the information on fines issued to members for the items that were borrowed and have become overdue at the time they were returned.

--Create Fine table


```

CREATE TABLE Fine (
  FineID int NOT NULL Primary Key,
  MemberID int NOT NULL Foreign Key
  (MemberID) References Members (MemberID),
  LoanID int NOT NULL Foreign Key
  (LoanID) References Loan (LoanID),
  OverdueFine money NULL);

```

Other tables involved in this database are the table for Payment which records information for payments made by members to the library for fines issued with balance outstanding and the table for Lost_Remove which keeps record of Items that are lost/removed from the library.

--Create Payment table

```

CREATE TABLE Payment (
  PaymentID int NOT NULL Primary Key,
  MemberID int NOT NULL Foreign Key
  (MemberID) References Members (MemberID),
  RepaymentDateTime DateTime NOT NULL,
  RepaymentAmount money NOT NULL,
  RepaymentMethod nvarchar(15) NOT NULL,
  OutstandingBalance money NULL);

```

--Create Lost_Removed table

```

CREATE TABLE Lost_Removed (
  Lost_RemovedID int NOT NULL,
  CatalogueID int NOT NULL Foreign Key
  (CatalogueID) References Catalogue (CatalogueID),
  DateIdentified Date NOT NULL,
  ISBN int NULL);

```

INSERT STATEMENTS

The following statements below would be necessary to insert values into the tables that have been created in the database in order to demonstrate to the client that all **SELECT** queries, user-defined functions, stored procedures, and triggers are working as expected.

(a) Insert into Catalogue table

```
INSERT INTO Catalogue (CatalogueID, ItemTitle, ItemType, Author, YearOfPublication, DateAdded, CurrentStatus)
VALUES ('101', 'CCN', 'Book', 'James Katre', '2009', '2014-02-22', 'OnLoan'),
('102', 'MA', 'Book', 'John Borrow', '2012', '2014-01-28', 'Available'),
('103', 'EXTC', 'Book', 'Keiron Rodge', '2011', '2013-11-05', 'Available'),
('104', 'IPMV', 'DVD', 'James Katre', '1999', '2021-10-18', 'OnLoan'),
('105', 'MA', 'Book', 'Brinn Mike', '2006', '2014-01-16', 'OnLoan'),
('106', 'ARWP', 'Journal', 'Nadi Hulburt', '2018', '2019-05-01', 'Available'),
('107', 'Mechanic', 'Other Media', 'Karan Simmit', '2020', '2022-06-01', 'OnLoan'),
('108', 'Electra', 'Book', 'John Borrow', '2021', '2023-01-29', 'Available');
```

(b) Insert into the Members table

```
INSERT INTO Members (MemberID, FirstName, LastName, Address1, Address2, City, Postcode, DateOfBirth, Username, Password, Email, Telephone, DateJoined, EndDate)
VALUES ('01', 'Rustie', 'Pandey', '24562 Katie Alley', NULL, NULL, 'LN6 7HQ', '2003-01-26', 'Rusto', 'Rusti001', 'rpandey0@topsy.com', '445-608-4639', '2023-01-29', NULL),
('1002', 'Ingeborg', 'Tytherton', '6455 Rockefeller Street', NULL, NULL, 'SY4 4TK', '1970-05-01', 'Ineg11', 'dhgywiuru', NULL, NULL, '1994-08-25', NULL),
('1003', 'Garik', 'Rennock', '85208 Coolidge Plaza', NULL, NULL, 'CT16 0BU', '1984-12-04', 'GerickRen', 'xdxderhyf', 'grennock2@vimeo.com', '920-370-3890', '2008-06-21', NULL),
('1004', 'Brinn', 'Merrywether', '59 Derek Hill', NULL, NULL, 'LS6 6HQ', '1987-05-24', 'BrinNer', 'fuhufd', 'bmerrywether3@google.ru', '110-127-3396', '2005-09-13', NULL),
('1005', 'Stanislaus', 'Elsie', '9169 Springs Parkway', NULL, NULL, 'M34 12J', '1965-02-19', 'StainE1', 'ivdrbs', 'selsie4@hibu.com', '770-639-5396', '1990-10-29', '2020-02-11'),
('1006', 'Howard', 'Staning', '8022 Sundown Parkway', NULL, NULL, 'KM10 8LS', '1990-06-02', 'HowardSt', 'wdhygyudc', NULL, NULL, '2017-03-27', NULL),
('1007', 'Nedi', 'Harrinson', '268 Clarendon Street', NULL, NULL, 'DL8 4TS', '1982-04-18', 'NediH', 'Hjfhuos', NULL, NULL, '2006-12-08', NULL),
```

(c) Insert into the Loan table

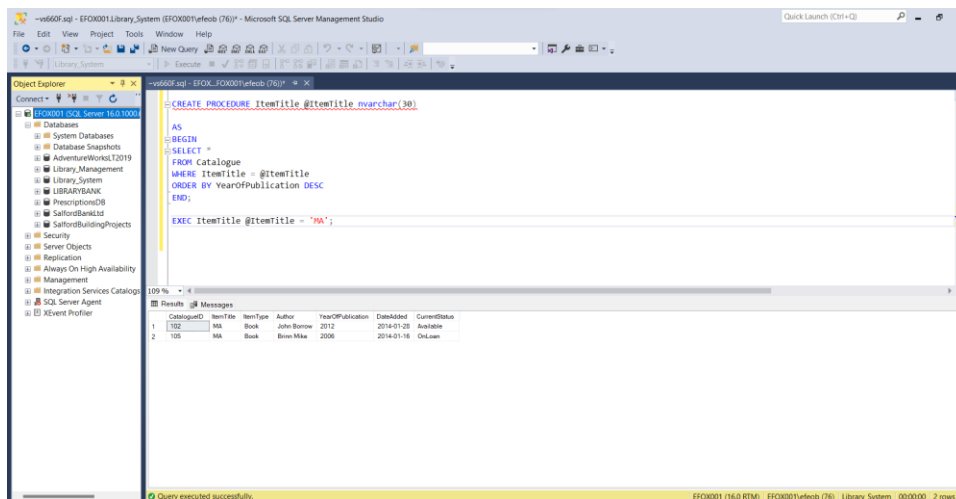
```
INSERT INTO Loan (LoanID, CatalogueID, MemberID, LoanDate, DueDate, ReturnDate, DaysOverdue)
VALUES ('2001', '102', '1003', '2022-12-17', '2022-12-26', '2023-01-02', '7'),
('2002', '104', '1003', '2023-03-22', '2023-03-31', NULL, NULL),
('2003', '101', '1010', '2023-03-21', '2023-03-30', NULL, NULL),
('2004', '107', '1005', '2023-02-15', '2023-02-24', '2023-02-28', '4'),
('2005', '107', '1002', '2023-03-02', '2023-03-11', '2023-03-08', NULL),
('2006', '105', '1212', '2023-03-20', '2023-03-29', NULL, NULL),
('2007', '106', '1010', '2023-02-19', '2023-02-28', '2023-02-24', NULL),
('2008', '108', '1009', '2022-12-28', '2023-01-06', '2023-01-05', NULL);
```

(d) Insert into the Fine table

```
INSERT INTO Fine (FineID, MemberID, LoanID, OverdueFine)
VALUES ('4001', '1003', '2001', '70'),
('4002', '1005', '2004', '40');
```

2. STORED PROCEDURES OR USER DEFINED FUNCTIONS

(a) Searching the Catalogue for matching character strings, and sorting the results by most recent publication date first which allows the query of the Catalogue looking for a specific item.



(b) Return a full list of all items currently on loan which have a due date of less than five days from the current date

(c) Inserting a member into the database, before execution the Members table is as displayed below

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'EFOX001'. The main query window shows the result of a 'SELECT * FROM Members' query. The Results pane displays a table with 12 columns and 9 rows of data.

MemberID	FirstName	LastName	Address1	Address2	City	Postcode	DateOfBirth	UserName	Password	Email	Telephone	DateJoined	EndDate
1	Phoebe	Pender	2452 Kula Alley		NULL	NULL	1963-01-26	Phoebe1	Phoebe1	phoebe1@harry.com	445-455-4555	2023-01-26	NULL
2	1002	Ingenborg	6425 Ruckelshier Street		NULL	514 47K	1970-05-01	ing11	dghw11	NULL	NULL	1994-06-29	NULL
3	1023	Gark	91228 Conlidge Place		NULL	CT16 024	1984-12-24	DanielPaw	adash17	qemmk2@harry.com	620-370-3880	2009-06-21	NULL
4	1004	Born	Meynether	50 Dewk Hill	NULL	NULL	1981-05-24	Borniller	huhd	bonyweeth3@poozie.ru	110-121-3366	2009-06-13	NULL
5	1005	Stanislas	Elae	9189 Springs Parkway	NULL	MS4 12J	1985-02-19	Stan01	adash	satash4@hnu.com	775-639-6366	1980-10-29	2020-02-11
6	1009	John	Mike	1234 Crown Avenue	NULL	MA01 782	1975-12-29	John01	hewer	johndead@hnu.com	777-666-6666	1982-12-09	2023-03-22
7	1010	Nadi	Harrison	1280 Cleveland Street	NULL	DL8 4UG	1992-04-18	Nadi01	Hfhuus	Nadi4@hnu.com	777-666-6628	2006-12-08	NULL
8	1011	Perry	Tender	22 Lupton Avenue	Leeds	West Yorkshire	1984-05-24	Jedings	Awaw001	Jedings01@yahoo.com	NULL	1989-02-18	NULL
9	1212	Gerry	Hubert	7831 Lelandwood Point	NULL	872 2N6	1988-11-04	Gerrah	hhuury	ghubert@harry.com	388-622-6466	2009-06-21	NULL

Creating the stored procedure,

The screenshot shows the Microsoft SQL Server Management Studio interface. The main query window contains the SQL code to create a stored procedure named 'INSERTMember'. The Messages pane at the bottom shows the successful execution of the query.

```

CREATE PROCEDURE INSERTMember (
    @MemberID int, @FirstName nvarchar(20), @LastName nvarchar(20), @Address1 nvarchar(50), @Address2 nvarchar(50), @City nvarchar(50), @Postcode nvarchar(10),
    @DateOfBirth Date, @UserName nvarchar(30), @Password nvarchar(20), @Email nvarchar(100), @Telephone int, @DateJoined Date, @EndDate Date)
AS
BEGIN
    INSERT INTO Members
    VALUES (@MemberID, @FirstName, @LastName, @Address1, @Address2, @City, @Postcode, @DateOfBirth, @UserName, @Password, @Email, @Telephone,
    @DateJoined, @EndDate)
END;

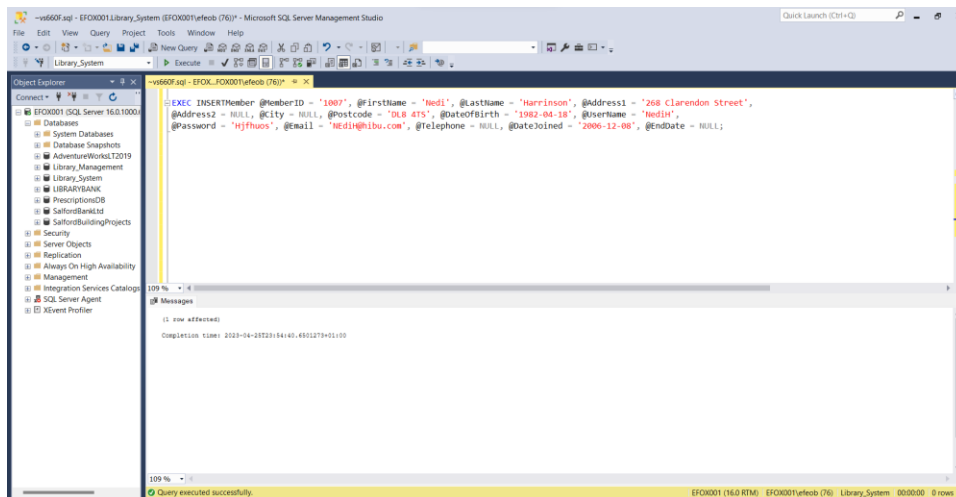
```

Messages

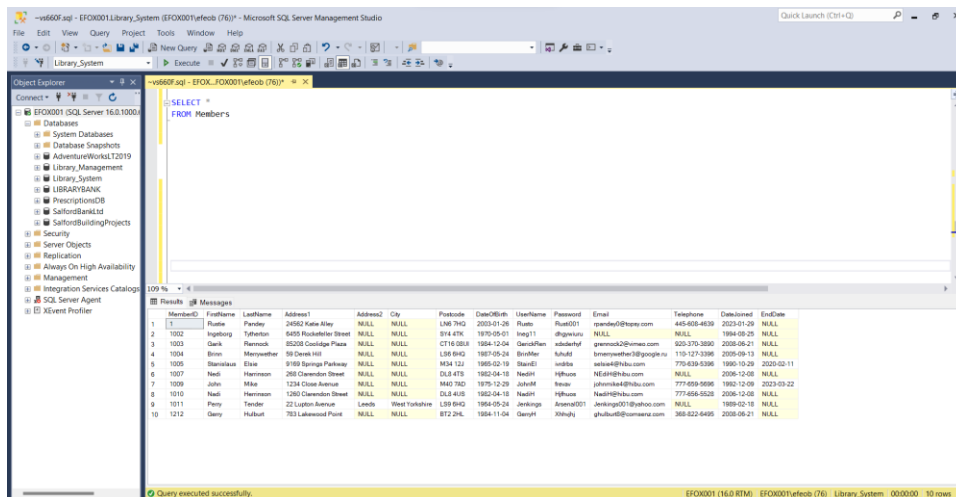
Command completed successfully.

Completion time: 2023-04-20T23:26:01.1440867+01:00

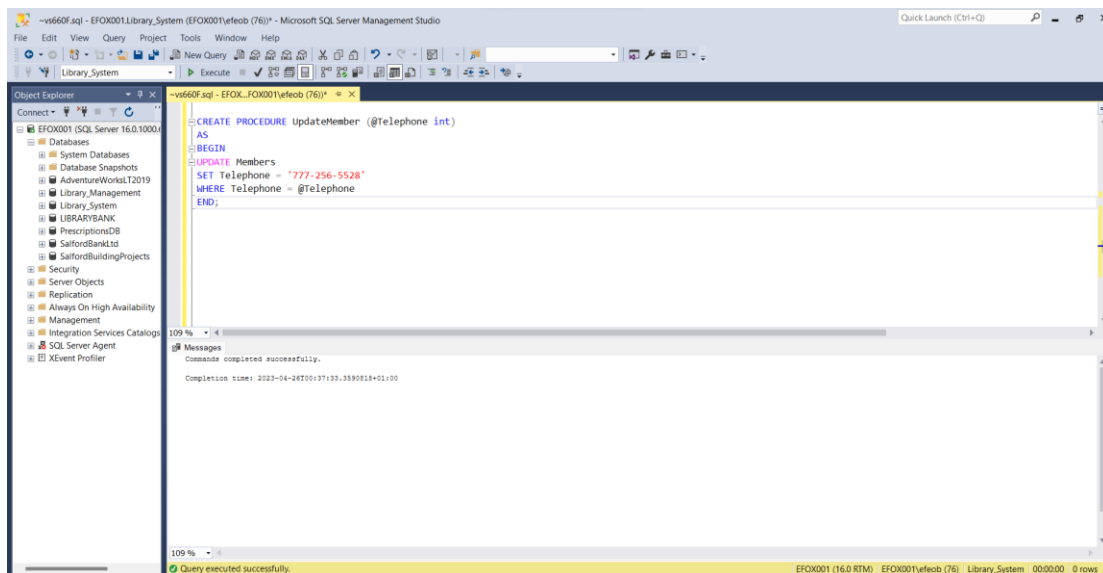
Executing the stored procedure,



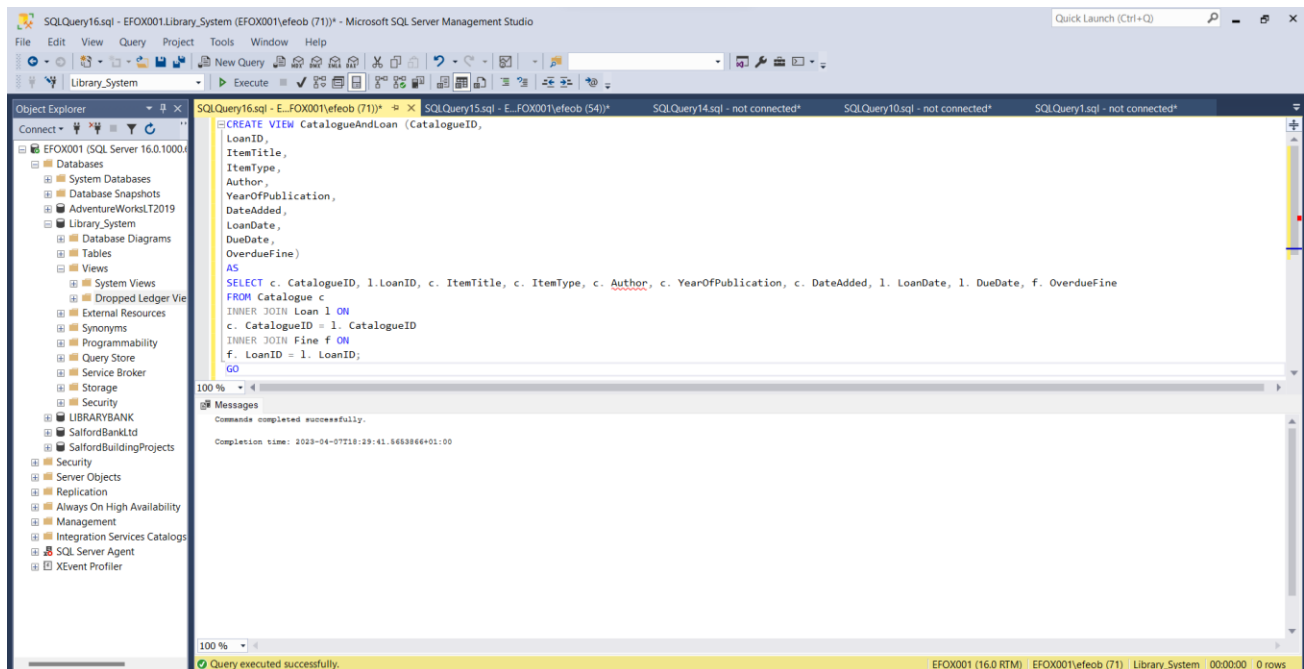
After executing the stored procedure,



(d) Creating the Stored procedure to updated the details of an existing member.



3. To view the loan history, showing all previous and current loans, and including details of the item borrowed, borrowed date, due date and any associated fines for each loan. See the Statement as displayed below



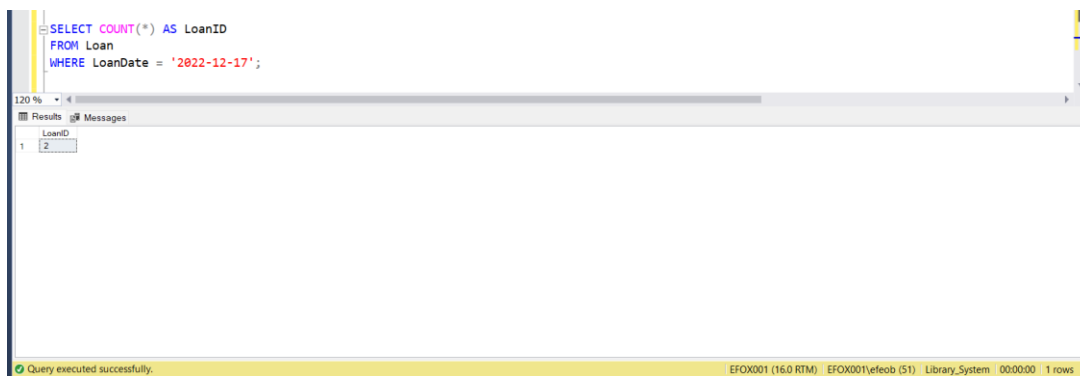
4. Create a trigger so that the current status of an item automatically updates to Available when the book is returned.

5. Below is a **SELECT** query which allows the library to identify the total number of loans made on a specified date. But before that I would make an **UPDATE** on the **Loan** table to ensure that the report is better presented.

UPDATE on **Loan** table as displayed below,

```
UPDATE Loan
SET LoanDate = '2022-12-17'
WHERE LoanDate = '2022-12-28';
```

Now the **SELECT** query as displayed below,



The screenshot shows a database query interface. The query editor at the top contains the following SQL query:

```
SELECT COUNT(*) AS LoanID
FROM Loan
WHERE LoanDate = '2022-12-17';
```

Below the query editor, the results are displayed in a table. The table has one column labeled 'LoanID' and one row with the value '2'. The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

LoanID
2

RECOMMENDATIONS

(a). **Data integrity and concurrency:** Concurrency is the ability of the DBMS to process more than one transaction at a time. The database manager must allow for access to the resources of the database by multiple interactive users while ensuring that the data integrity is preserved. The database manager controls this access to prevent undesirable effects, such as:

- **Lost updates.** Two applications, A and B, might both read the same row and obtain new values for one of the columns based on the data that these applications read. If A updates the row and then B also updates the row, A's update is lost.
- **Access to uncommitted data.** Application A might update a value, and B might read that value before it is committed. Then, if A backs out of that update, the calculations performed by B might be based on invalid data.
- **Non-repeatable reads.** Application A might read a row before processing other requests. In the meantime, B modifies or deletes the row and commits the change. Later, if A attempts to read the original row again, it sees the modified row or discovers that the original row has been deleted.
- **Phantom reads.** Application A might execute a query that reads a set of rows based on some search criterion. Application B inserts new data or updates existing data that would satisfy application A's query. Application A executes its query again, within the same unit of work, and some additional ("phantom") values are returned.

To control concurrency in the database, the database manager must:

- (i) enforce isolation among transactions.
- (ii) preserve the consistency of the database through consistently preserving the execution of transactions.
- (iii) resolve read-write and write-read conflicts.

(b). Database security: This implies keeping sensitive information safe and preventing the loss of data. Security of database is controlled by the Database Administrator (DBA).

The following are the main control measures used to provide security of data in databases:

1. Authentication
2. Access control
3. Inference control
4. Flow control
5. Database Security applying Statistical Method
6. Encryption

These are explained as following below.

I. Authentication:

Authentication is the process of confirmation that whether the user log in only according to the rights provided to him to perform the activities of data base. A particular user can login only up to his privilege, but he can't access the other sensitive data. The privilege of accessing sensitive data is restricted by using Authentication. By using these authentication tools for biometrics such as retina and figure prints can prevent the database from unauthorized/malicious users.

II. Access

Control:

The security mechanism of DBMS must include some provisions for restricting access to the database by unauthorized users. Access control is done by creating user accounts and controlling login process by the DBMS. So, database access of sensitive data is possible only to those people (database users) who are allowed to access such data and to restrict access to unauthorized persons. The database system must also keep track of all operations performed by certain users throughout the entire login time.

III. Inference

Control:

This method is known as the countermeasures to statistical database security problems. It is used to prevent the user from completing any inference channel. This method protects sensitive information from indirect disclosure. Inferences are of two types, identity disclosure or attribute disclosure.

IV. Flow

Control:

This prevents information from flowing in a way that it reaches unauthorized users. Channels are the pathways for information to flow implicitly in ways that violate the privacy policy of a company and are called covert channels.

V. Database Security applying Statistical Method:

Statistical database security focuses on the protection of confidential individual values stored in and used for statistical purposes and used to retrieve the summaries of values based on categories. They do not permit us to retrieve the individual information. This allows access to the database to get statistical information about the number of members in the library but not to access the detailed confidential/personal information about the specific member.

VI. Encryption:

This method is mainly used to protect sensitive data (such as library card numbers) and other sensitive numbers. The data is encoded using some encoding algorithms. An unauthorized user who tries to access this encoded data will face difficulty in decoding it, but authorized users are given decoding keys to decode data.

(c.) Database backup and recovery: The backup and recovery of data is the process of backing up your data in the event of a loss and setting up secure systems that allow you to recover your data as a result. Data backup requires the copying and archiving of computer data to make it accessible in case of data corruption or deletion. You can only recover data from an earlier time if you have backed it up with a reliable backup device.

Data backup is one form of **disaster recovery** making it an essential part of any sensible disaster recovery plan.

Today, you can back up a significant deal of data using cloud storage therefore, archiving your data on a local system's hard drive or external storage is not necessary. What's more, you can set up your devices using cloud technologies to allow automatic data recovery.

Conclusion

The library management system is essential for all communities, colleges, schools, and many more places. A lot of manual work can be reduced with this library management system. Also, a lot of glitches like wrong borrow date and miscalculation of fine amount are avoided. As it is a computer-managed system and so these are all avoided. It is also efficient and cost-effective. This Library management system stores the details of items like Book, Journal, DVD or Other Media and also details of library members. So overall, we have seen-

- The build-up of a database to maintain all related information.
- The build-up of tables separately to store data.
- Learned the purpose of the library management system.
- What features are required for library members and librarians to use LMS?
- How the software allows storing of all the details related to the library.
- Finally, we tested the final database.

TASK 2

Database Design & SQL for Data Analysis

Introduction

Information has been the key to a better organization and new developments. The more information we have, the more optimally we can organize ourselves to deliver the best outcomes. That is why data collection is an important part for every organization. We can also use this data for the prediction of current trends of certain parameters and future events. By introducing database design and sql for data analysis on a real-world dataset released every month by the National Health Service (NHS) in England it is possible to analyse the prescribing data to understand more about the types of medication being prescribed, the organisations doing the prescribing, and the quantities prescribed, and this information can be used to develop, use and balance the different types of KPIs for strategic and operational improvement.

Key performance indicators (KPIs) are metrics that evaluate the outcomes of an organization, or of particular activities. The assessment of performance identifies gaps between current and desired outcomes and provides an indication of progress towards closing the gaps. KPIs are commonly used to measure and improve upon the performance of healthcare systems; however, no KPIs have been specifically developed to evaluate or compare prescription drug systems across jurisdictions.

The National Health Service (NHS) accounts for more than 98% of the UK prescription medicines market, which is the sixth largest pharmaceutical market in the world. Most of this market is driven by the UK's approximately 35,000 general practitioners (GPs). It is an open market, with most leading foreign pharmaceutical companies having a strong presence. While the growth rate of this market has been decelerating, it remains one of the fastest growing components of NHS expenditure. The NHS does not operate any kind of national reimbursement list, but the UK government has adopted several means to keep medicines expenditure under control. These include cash incentives and constraints for GPs relating to expenditure on

medicines, individual quarterly updates on GP prescribing, the publication of a list of medicines that cannot be prescribed by GPs, the switching of some prescription-only medicines to over-the-counter medicines, and a co-payment system. The main form of economic regulation in the UK, however, remains the Pharmaceutical Price Regulation Scheme (PPRS). This limits the rate-of-return on capital attributable to medicines sales to the NHS, with the intended rate-of-return being equal to that of UK industry overall. The pharmaceutical industry has generally performed relatively well in the UK market, managing to preserve incentives to innovation. This reflects the fact that UK GPs have been able to maintain their clinical freedom, as well as government recognition of the economic contribution made by the pharmaceutical industry.

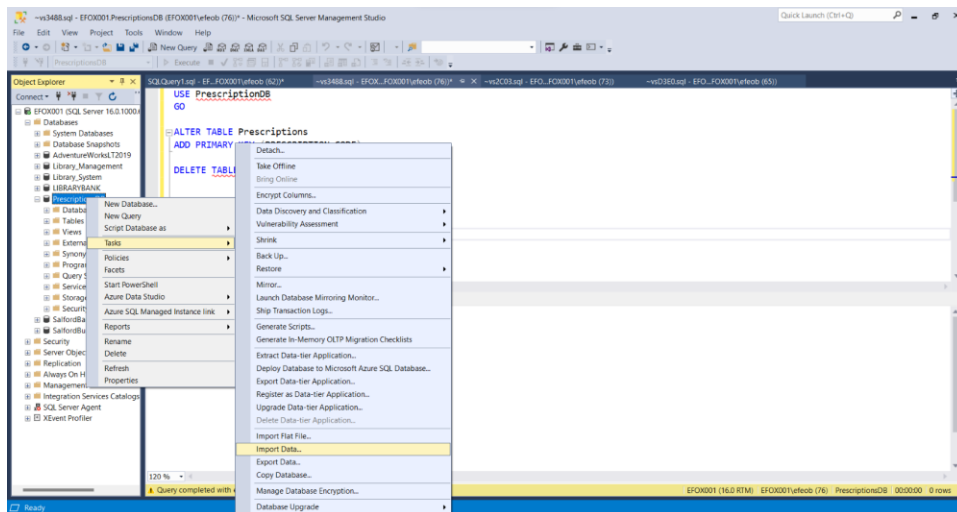
Current issues of interest in the UK pharmaceutical market context include the future of the PPRS, the debates over the imposition of a national formulary and generic substitution, and over parallel trade, the potential impact of managed-care protocols and computer-based prescribing on pharmaceutical expenditures, and possible political changes. Hence the need for a thorough data analysis of the prescription dataset in order to develop key performance indicators for prescription medication systems.

1. THE DATABASE DESIGN

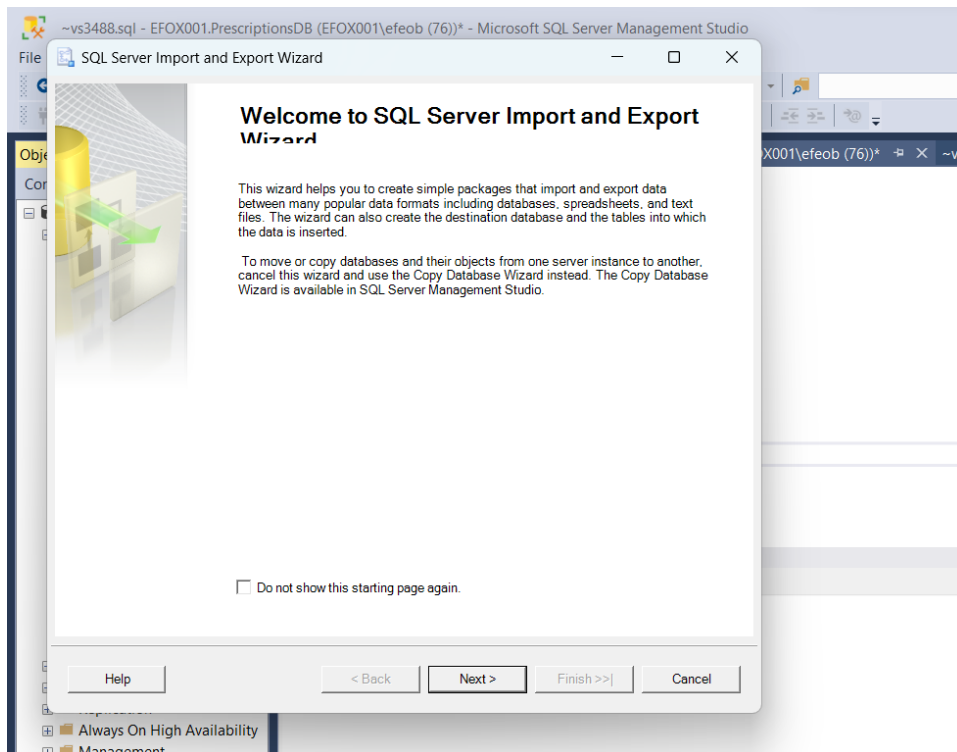
The first stage of this task has been to create a database with the name PrescriptionsDB. This is done on the Microsoft sql server management studio using the 'CREATE DATABASE PrescriptionsDB' statement.

The next would be the uploading of the associated tables into this database. The name of the tables are Drugs, Medical_Practice, and Prescriptions. These are the tables that make up the database 'PrescriptionsDB' and are currently stored in the local storage. This process of uploading involves different steps which would be analysed with the aid of the screenshots displayed below.

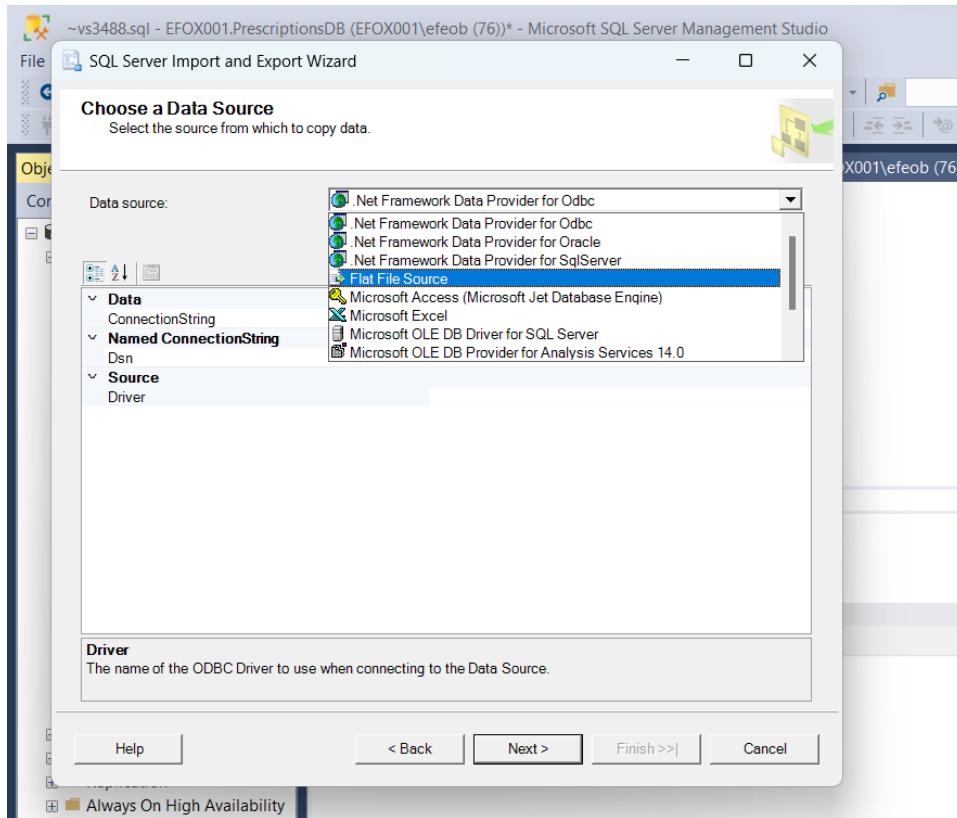
First in this case is right clicking on the database 'PrescriptionDB' and finding your way to import data as displayed.



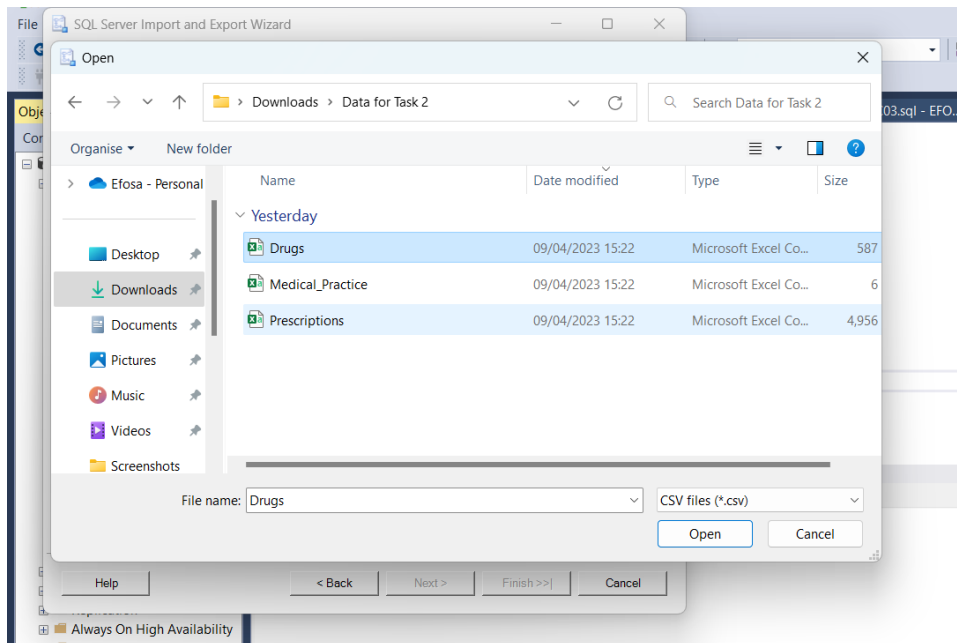
On the next stage, just click on next,



Next would be choosing the data source which is the flat file source and click next,

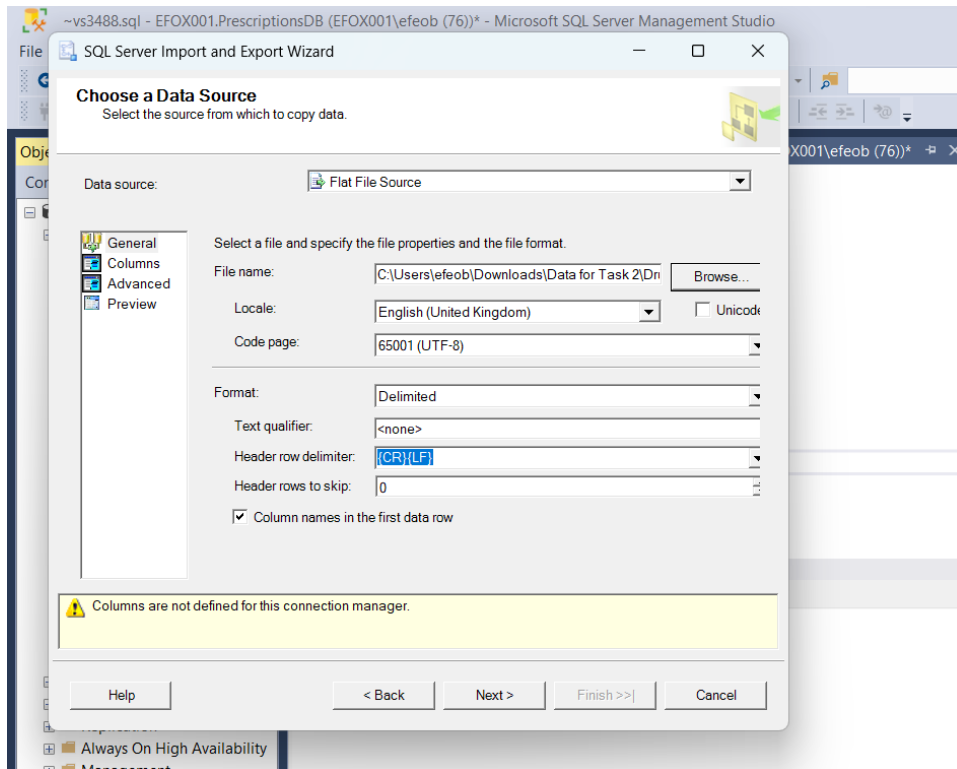


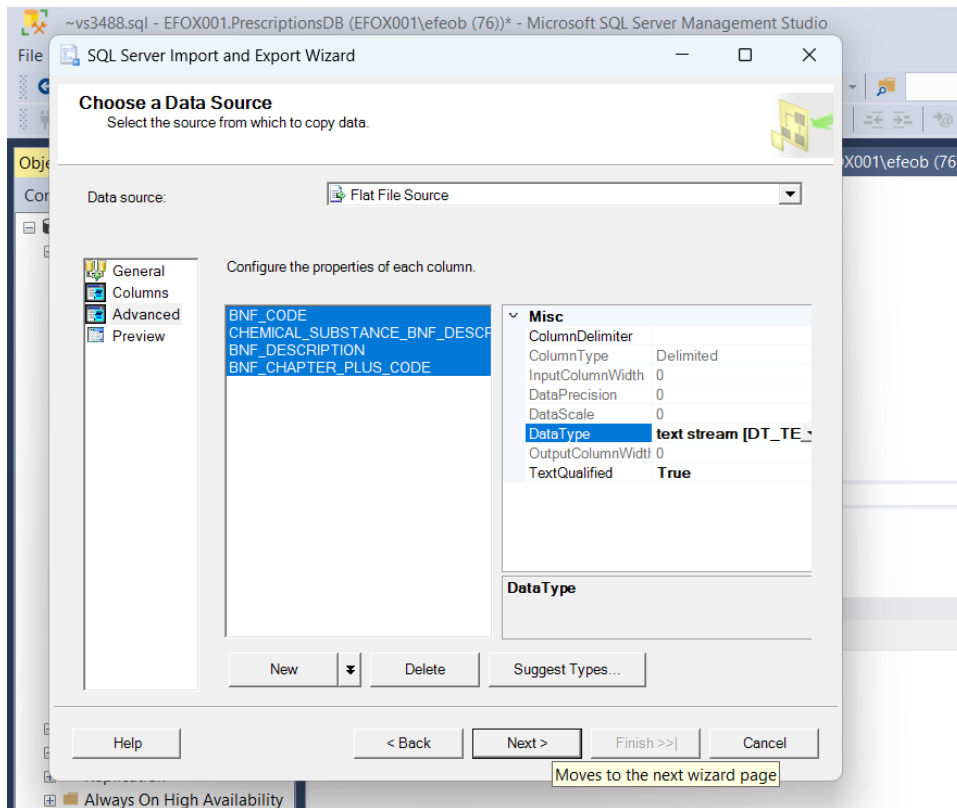
Clicking Next takes us to the location of our desire files for upload,



Click on the file to upload one at a time and then open the file using the .csv format and this process will be followed in uploading each of the other files in this storage location.

The next process now involves choosing the data types for the columns within the tables in this file and on this section of the data source, Click on **Advanced**, highlight the columns of the table that is displayed, then move to the **Data type** and select your desired data type for these columns. Then click on **next**





On concluding with the stage displayed above, click on next, next again, close and then finish.

The three different tables of the database 'PrescriptionsDB' have now been successfully uploaded into the database. So, the next thing to do would be to define and establish the relationship among the tables in the database. I would start by creating the primary and the foreign keys using the 'ALTER' Statement .

For the table Medical_Practice, the primary key is added to the column 'PRACTICE_CODE' by first altering the data type to varchar and establishing the constraint.

```
ALTER TABLE Medical_Practice
ALTER COLUMN PRACTICE_CODE varchar(15) NOT NULL
```

```
ALTER TABLE Medical_Practice
ADD PRIMARY KEY (PRACTICE_CODE)
```

This process is repeated for the table 'Drugs' using the column 'BNF_CODE' for the primary key and varchar(50) for the data type.

The Statement below adds the primary key to the column PRESCRIPTION_CODE by first altering the data type to integer and establishing the constraint.

```
ALTER TABLE Prescriptions  
ALTER COLUMN PRESCRIPTION_CODE int NOT NULL
```

```
ALTER TABLE Prescriptions  
ADD PRIMARY KEY (PRESCRIPTION_CODE)
```

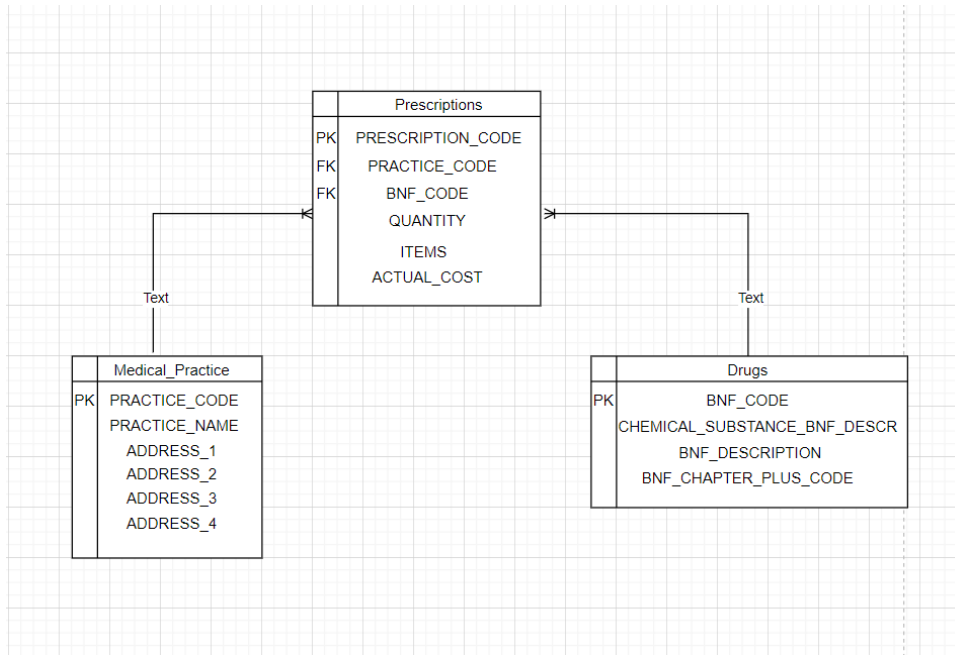
Next is adding foreign keys to both columns 'PRACTICE_CODE' and 'BNF_CODE' in the Prescription table.

```
ALTER TABLE Prescriptions  
ADD FOREIGN KEY (PRACTICE_CODE) REFERENCES Medical_Practice (PRACTICE_CODE)
```

```
ALTER TABLE Prescriptions  
ADD FOREIGN KEY (BNF_CODE) REFERENCES Drugs (BNF_CODE)
```

Now the relationship that exist among the tables in the database 'PrescriptionsDB' using the 'ALTER' Statement has been established and the next would be defining their relationship using the ER DIAGRAM.

ER DIAGRAM



This is the entity relational diagram of the database 'PrescriptionsDB', this diagram displays the tables within the database and the relationship that exist among them. The line connecting the tables 'Medical_Practice' and 'Drugs' to the Prescriptions table is called 'one to many'. It indicates that one PRACTICE_CODE can be associated to more than one PRESCRIPTION_CODE and one BNF_CODE can be associated to more than one PRESCRIPTION_CODE as well.

2. Details of all drugs which are in the form of tablets or capsules.

Query Results:

BNF_DESCRIPTION
Magnesium oxide 100mg tablets
Ornithine 100 capsules
Sildenafil 100mg capsules
Sildenafil 100mg capsules
Woodblock 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules
Ramipril 100mg capsules

3. The total quantity for each of prescriptions

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL query:

```
SELECT COUNT(pc.PRESCRIPTION_CODE) AS PRESCRIPTION_CODE_COUNT, d.BNF_CHAPTER_PLUS_CODE, (AVG(CAST(ACTUAL_COST AS MONEY))) AS AVG_COST,
(MIN(CAST(ACTUAL_COST AS MONEY))) AS MIN_COST, (MAX(CAST(ACTUAL_COST AS MONEY))) AS MAX_COST
FROM Prescriptions pc
INNER JOIN Drugs d ON
d.BNF_CODE = pc.BNF_CODE
GROUP BY BNF_CHAPTER_PLUS_CODE
```

The Results pane displays the following data:

	PRESCRIPTION_CODE_COUNT	BNF_CHAPTER_PLUS_CODE	AVG_COST	MIN_COST	MAX_COST
1	1	10mm ruler 20 Drawings	143.0362	143.0362	143.0362
2	1	50-80mm diameter, polyether 21. Appliances	2.2376	2.2376	2.2376
3	3	50-80mm diameter, PVC 21. Appliances	4.941	2.4248	4.9402
4	1	57mm diameter, silicone without supporting 21...	20.8807	20.8807	20.8807
5	2	56mm diameter, silicone without supporting 21...	31.321	20.8807	41.7614
6	2	70mm diameter, silicone with supporting 21. Ap...	17.4484	14.0181	20.8807
7	1	70mm diameter, silicone without supporting 21...	20.8807	20.8807	20.8807
8	1	70mm diameter, silicone with supporting 21. Ap...	20.8807	20.8807	20.8807
9	1	70mm diameter, silicone without supporting 21...	20.8807	20.8807	20.8807
10	2	63mm diameter, silicone without supporting 21...	17.4484	14.0181	20.8807
11	1	55-100mm diameter, PVC 21. Appliances	4.9402	4.9402	4.9402
12	1	with border & enhanced tap 15-55mm 23. Stoma Ap...	194.0696	194.0696	194.0696
13	1	with border & enhanced tap 15-55mm 23. Stoma Ap...	194.0696	194.0696	194.0696
14	1	adhesive border & enhanced tap 15-30mm beige 2...	64.5406	64.5406	64.5406
15	1	CeraPlus skin & Lock 'n' Roll closure, max 15-25m...	204.6391	204.6391	204.6391
16	1	CeraPlus skin & Lock 'n' Roll closure, max 15-51m...	142.3196	142.3196	142.3196

The status bar at the bottom indicates: Query executed successfully. EFOX001 (16.0 RTM) EFOX001\efox001 (52) PrescriptionsDB 00:00:00 232 rows

6. The most expensive prescription prescribed by each practice, sorted in descending order by prescription cost. Returning only those rows where the most expensive prescription is more than £4000. Including the practice name in your result.

7. Five queries with a brief explanation.

i. List of the total amount of prescriptions issued and the overall cost by each medical practice.

The screenshot shows a SQL query in the 'SQLQuery1.sql' window. The query is as follows:

```
SELECT m. PRACTICE_CODE, COUNT (pc. PRESCRIPTION_CODE) AS TOTAL_PRESCRIPTIONS, (SUM (CAST(ACTUAL_COST AS MONEY))) AS AMOUNT_SPENT
FROM Prescriptions pc
INNER JOIN Medical_Practice m ON
m. PRACTICE_CODE = pc. PRACTICE_CODE
GROUP BY m. PRACTICE_CODE
```

The results are displayed in the 'Results' pane, showing a table with three columns: PRACTICE_CODE, TOTAL_PRESCRIPTIONS, and AMOUNT_SPENT. The table contains 17 rows of data.

PRACTICE_CODE	TOTAL_PRESCRIPTIONS	AMOUNT_SPENT
PD2023	2218	69379.1621
PD2052	1709	47406.824
PD2009	1830	61188.0206
YD2790	2104	66274.0422
YD2641	212	15001.1031
PD2009	2879	108554.5499
PD2030	1556	47382.0181
PD2054	1	1.5000
PD2025	880	28543.2732
PD2018	2342	62049.2862
PD2007	4189	202296.3469
PD2012	2237	84490.8547
PD2003	3821	183615.834
PD2004	3058	121360.4432
PD2013	2527	82529.649
PD2029	1213	28113.1013
DD16104	1763	69100.5581

The status bar at the bottom indicates: EFOX001 (16.0 RTM) EFOX001\efeb (53) PrescriptionsDB 00:00:00 60 rows.

ii. Select all drugs the has hydrochloride in its chemical composition.

The screenshot shows a SQL query in the 'SQLQuery1.sql' window. The query is as follows:

```
SELECT BNF_DESCRIPTION
FROM Drugs
WHERE CHEMICAL_SUBSTANCE_BNF_DESCR LIKE "Hydrochloride"
```

The results are displayed in the 'Results' pane, showing a table with one column: BNF_DESCRIPTION. The table contains 16 rows of data.

BNF_DESCRIPTION
Disopiramine 10mg/5ml oral solution
Disopiramine 10mg tablets
Disopiramine 20mg tablets
Rabeprazole gel
Mefenamic 50mg/5ml oral suspension sugar free
Mefenamic 150mg tablets
Mefenamic 200mg modified release capsules
Citalopram 10mg tablets
Citalopram MR 200mg capsules
Hydrochloric acid 3.6g Mefenamic 150mg effervescent granules sachets
Paracetamol Mefenamic effervescent granules sachets
Loperamide 2mg suppositories
Loperamide 1mg/5ml oral solution sugar free
Loperamide 2mg tablets
Loperamide 2mg modified release tablets sugar free
Modium Original 2mg capsules

The status bar at the bottom indicates: EFOX001 (16.0 RTM) EFOX001\efeb (53) PrescriptionsDB 00:00:00 583 rows.

iii. List of the total amount of prescriptions issued and the overall cost by each medical practice with the overall cost greater than 80000 and the total amount of prescriptions in the order of max to min.

SQLQuery4.sql - EFOX001PrescriptionsDB (EFOX001)efeb (52) - Microsoft SQL Server Management Studio

```

SELECT m. PRACTICE_CODE, COUNT (pc. PRESCRIPTION_CODE) AS TOTAL_PRESCRIPTIONS, (SUM (CAST(ACTUAL_COST AS MONEY))) AS AMOUNT_SPENT
FROM Prescriptions pc
INNER JOIN Medical_Practice m ON
m. PRACTICE_CODE = pc. PRACTICE_CODE
GROUP BY m. PRACTICE_CODE
HAVING (SUM (CAST(ACTUAL_COST AS MONEY))) > 80000
ORDER BY TOTAL_PRESCRIPTIONS DESC

```

PRACTICE_CODE	TOTAL_PRESCRIPTIONS	AMOUNT_SPENT
PB2015	4977	333009.637
Y03076	4355	227230.5236
PB2007	4189	202286.3469
PB2008	3920	190090.5747
PB2003	3821	193615.634
PB2001	3695	158031.5961
PB2016	3676	170756.1767
PB2006	3479	141759.7048
PB2031	3377	150511.846
PB2002	3173	117579.8895
PB2004	3058	121362.4432
PB2009	2879	10584.5499
PB2014	2856	108978.056
PB2021	2856	108049.7897
PB2010	2850	122959.8064
PB2023	2783	132516.1116
BH7006	1662	155553.6166

Query executed successfully. EFOX001 (16.0 RTM) EFOX001efeb (52) PrescriptionsDB 00:00:00 22 rows

iv. Nested query including use of IN

SQLQuery5.sql - EFOX001PrescriptionsDB (EFOX001)efeb (57) - Microsoft SQL Server Management Studio

```

SELECT PRESCRIPTION_CODE, BNF_CODE, QUANTITY, ITEMS, ACTUAL_COST
FROM Prescriptions
WHERE PRACTICE_CODE IN (
SELECT PRACTICE_CODE
FROM Medical_Practice
WHERE PRACTICE_NAME LIKE '%SURGERY%');

```

PRESCRIPTION_CODE	BNF_CODE	QUANTITY	ITEMS	ACTUAL_COST
5162	20090200130	2	1	7.35813
5163	20090200159	10	1	34.90204
5164	20090200159	40	1	139.60817
5165	20090900022	30	1	96.48783
5166	200909000455	1	1	1.07825
5167	200909000457	1	1	1.13435
5168	21010230115	1	3	14.42623
5169	21010230116	1	18	144.11349
5170	21010230116	2	2	32.00042
5171	21010230177	1	1	6.13639
5172	21010230178	1	6	37.0427
5173	210109000448	100	1	14.41078
5174	210109000600	100	1	4.67795
5175	210109000603	100	1	4.67795
5176	210109000603	200	1	9.3433
5177	210109000607	90	3	10.10672
5178	210109000607	500	1	30.47604

Query executed successfully. EFOX001 (16.0 RTM) EFOX001efeb (57) PrescriptionsDB 00:00:00 26,990 rows

CONCLUSION

These indicators are recommended as a starting point to assess the current performance of prescription medication systems across different jurisdictions. Consideration should be given to developing indicators in

additional disease areas as well as indicators that measure the domains of timeliness and patient-centeredness. In order to understand differences—and improve performance within KPIs across jurisdictions—assessing the capacity of publicly funded medication systems will be important. Future work should focus on the feasibility of measuring these indicators.

