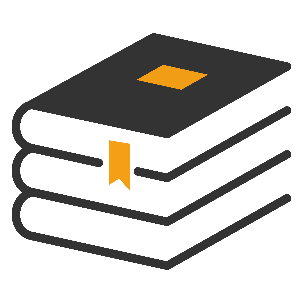
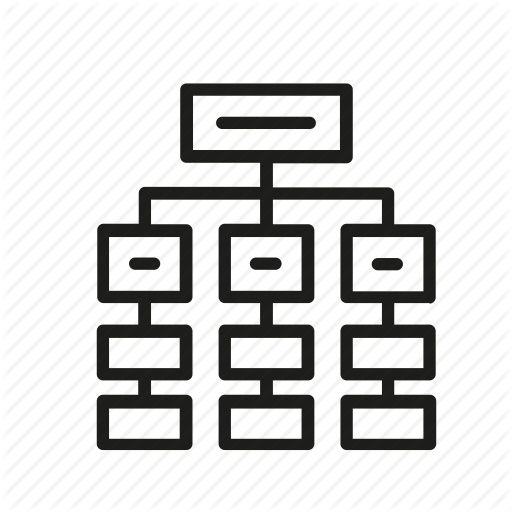
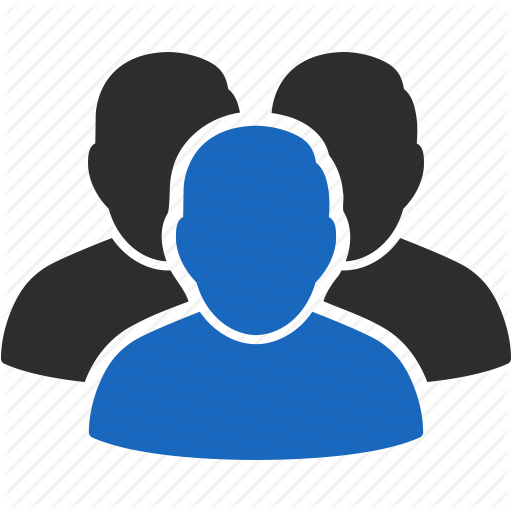
BOOKSTORE PROJECT

Ferhat Özkan – 150115009

Gökhan Çulfacı – 150113027

**About**

We aim to design a well organized, user friendly e-bookstore database which can be used in phone applications or web applications. To establish this goal we have to keep our database simple but in the same time it must be detailed and effective. Some features are searching a book by genre, searching for best sellers, searching for top trending books. Users will be able to order books and share their experience by voting them.

Depending on the feedback a book gets its rate will change. The average of all votings will be the rate which is displayed on the corresponding book details.

Also on every order of that book the system will remember the total number of orders. Which will make it able to the user to know the top sellers.

**Data and Requirement Analysis**

The database will consist of tables book table, user table, order table, payment type table, publisher table and author table.

Books are identified by an unique book id and their title, isbn, price, rate, sold quantity, publish date and genre must be recorded. A book may belongs to more than one genre. Rate can take values between 0-5 which is 0 for default and isbn must be a 13 digit number. Sold quantitiy is 0 as default.

Orders are identified by an unique order id and their order date, feedback must be recorded. Feedback can take values between 0-5.

Users are identified by an unique user id and first name, last name, phone number, address, email, birthdate and age must be recorded. Address is composed of city, state, and zipcode. Phone number must consist of 11 digits.

Authors are identified by an unique author id and first name, last name, birthdate must be recorded.

Every order has a payment type which includes a unique payment id. Also payment types have payment infos (cash, card etc.). Default value for cash is ‘Nakit’

Publishers are identified by an unique publisher id and their name, phone number, adress must be recorded. Address is composed of city, state, and zipcode. Phone number must consist of 11 digits.

A author can have more then one book but atleast one. A book can have only one author.

A book may be ordered many times. But every order can contain one book.

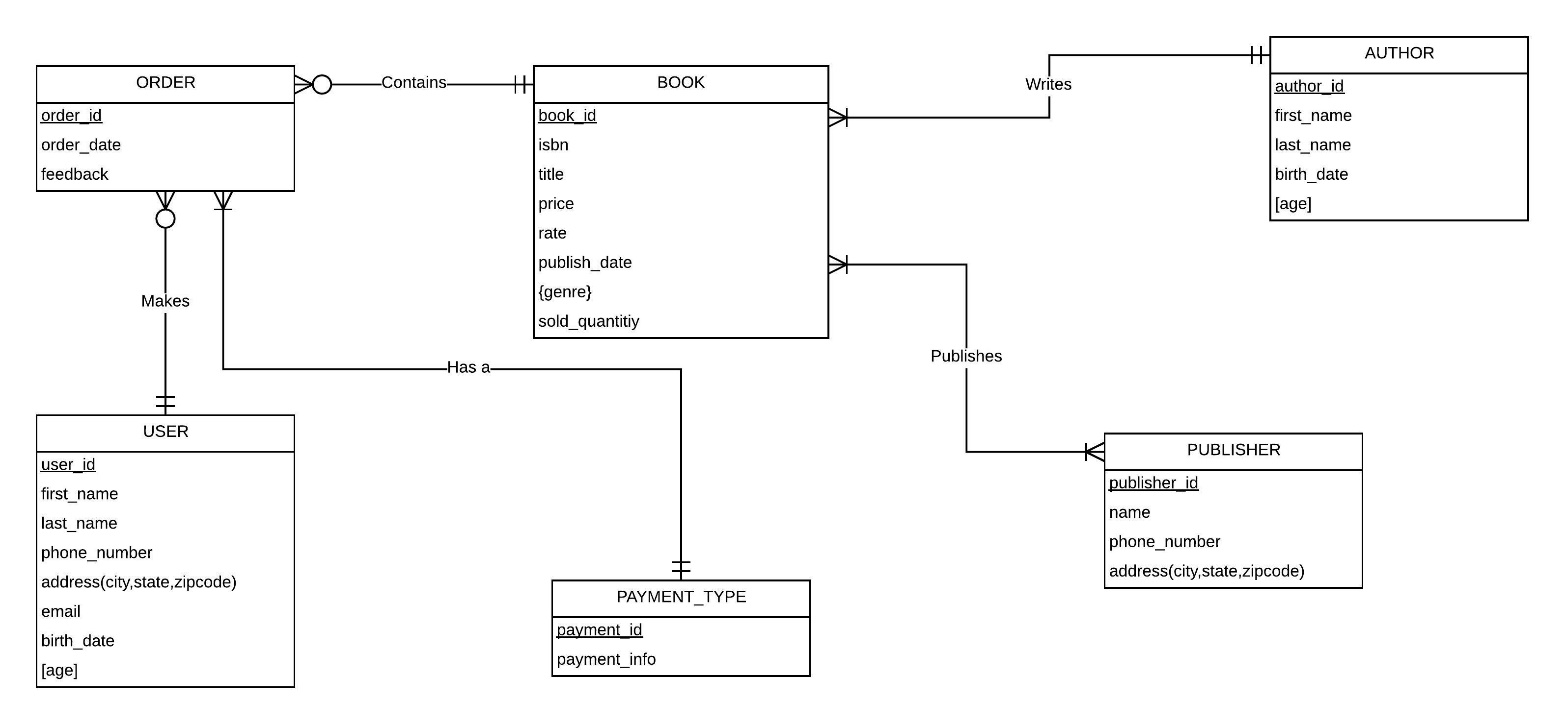
A user may have many orders. But every order must belong to a single user.

A order must be ordered with a single payment type.

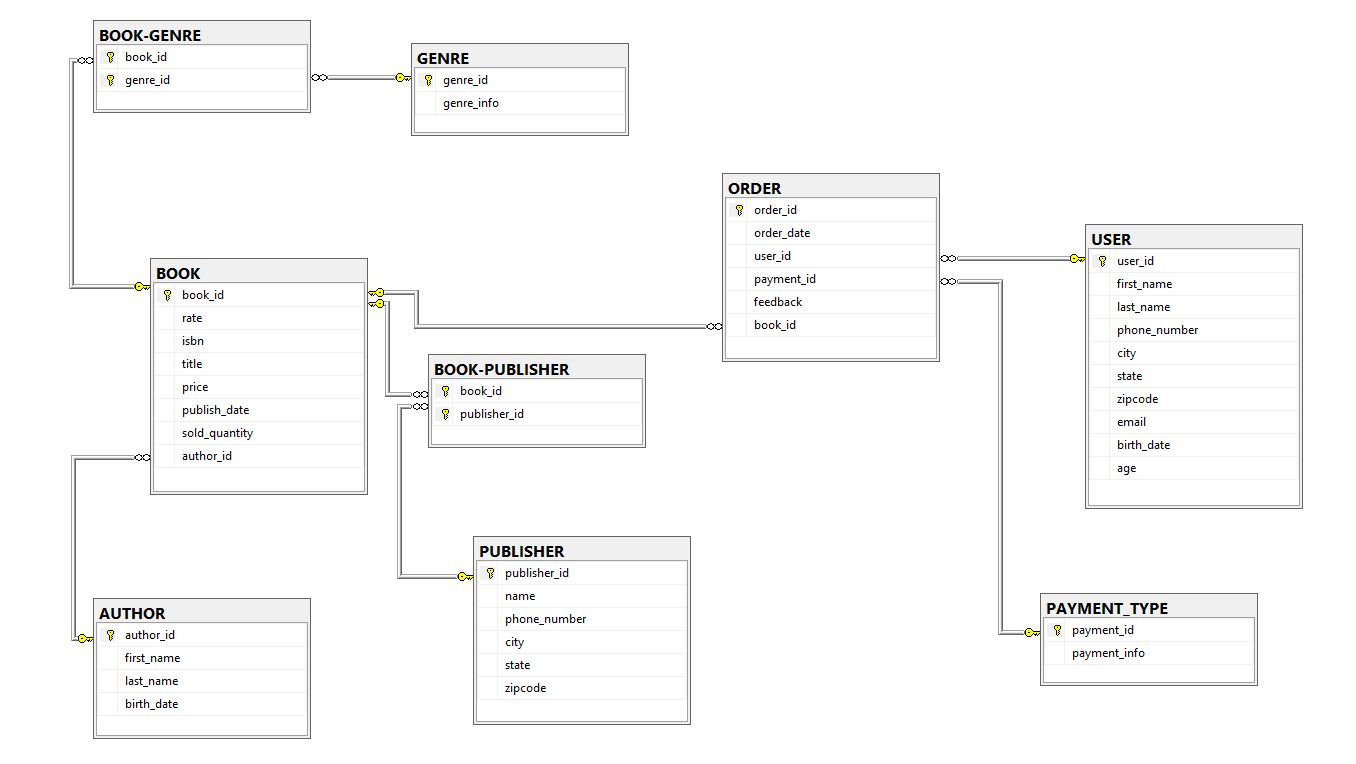
A book may published by many publishers. A publisher may publish many books.

For this project we have colloborated with the book store Mephisto kitabevi. Because of business privacy they didnt shared informations like the number of sells for every book but we could get a list of book names and their prices. We couldnt get a list of books for the project but they let us take photos and notes about the books informations.

**E-R Diagram**



**Database Diagram**



**Information About the Tables & Attiributes**

We created 9 tables for msql. These are Author, Book, Publisher,Payment\_Type, Order, Genre, User, Book-Publisher, Book-Genre. Now we can explain these tables :

* BOOK

We created Book table which contains all books registered in the database. The Book table has 8 attributes :

book\_id is primary key and unique. Every book has one book\_id.

Rate is avarage of all feedbacks and rate can take values between 1-5.

ISBN is an international standart book number. Every book has a unique isbn number.

Title is the title of books.

Price is the price of the books (decimal(19,4)). We have used this data type because a price can take values like 19.90.

Publish\_date shows date of publish for that book.

Sold quantity shows the number of orders for a corresponding book.

author\_id points to an author data in the author table who wrote the corresponding book.

* AUTHOR

We created Author Table which contains 4 attributes :

author\_id is a primary key and is unique. Every author has one unique author\_id.

Author also has first name , last name and birth\_date attributes. Birth date is stored in a special format date. (yy-mm-dd).

* PUBLISHER

We created Publisher table which contains 6 attributes:

publisher\_id is primary key and unique.Every publisher has one unique publisher\_id.

publisher also has name, city, state, zipcode.

phone number is unique for publishers.

* GENRE

We created Genre table. Genre shows the category of that book. Genre table contains two attiribute:

genre\_id is primary key and unique.

genre\_info is that takes nvarchar(50).(adventure,horror etc.)

* USER

We created User table because we will use these datas to create the UI for users. User table has 10 attributes:

first\_name, last\_name, city, state, zipcode .

email and phone\_number are unique.

age calculated from birth\_date. We have implemented a formula in the database. (CONVERT([smallint],datediff(year,[birth\_date],getdate()))).

* ORDER

We created Order table because we need store the orders. This table has 6 attributes:

order\_id is primary key and unique.

order date shows date of order. Which has a datatype date.

user\_id, book\_id and payment\_id are foreign keys.

feedback takes integer value between 1-5 for rate.

* PAYMENT\_TYPE

We created payment\_type table because every order can be ordered by two types of payment(cash or credit card). This table has 2 attributes:

payment\_id is primary key and unique.

payment info shows type of payment.

* BOOK-GENRE

We created Book-Genre table because a book can belong many genres and a genre can have many books.

This table has 2 foreign key :

book\_id and genre\_id.

* BOOK-PUBLISHER

We created Book-Publisher table because a book can be published by many publishers and obviously a Publisher publishes many boks. This table has 2 foreign key:

book\_id and Publisher\_id.

**CONSTRAINTS -- INDICES -- DEFAULTS :**

**Indices**

* We created indices for both uniqueness and primary keys : order\_id, book\_id, user\_id, Publisher\_id

**Constraints**

* We created check constraints for feedback , phone\_number and isbn.

We checked feedback from order table with this formula

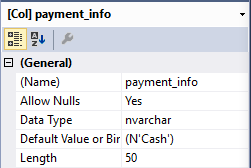
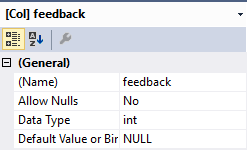
([feedback]>(0) AND [feedback]<(6))

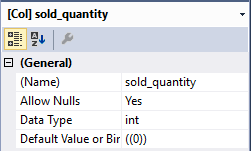
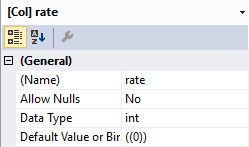
We checked phone numbers from Publisher and user tables with this formula (len([phone\_number])=(11)). This length of phone number is implemented for Turkey

We checked isbn from book table with this formula (len([isbn])=(13))

**Defaults**

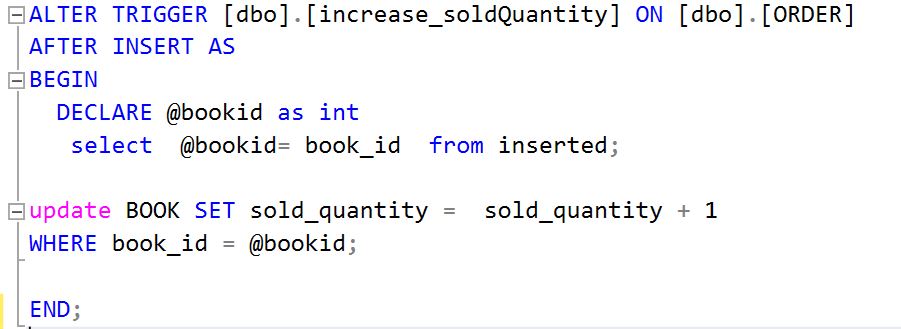
* We implemented defaults value for payment\_info, sold\_quantity, rate, feedback.

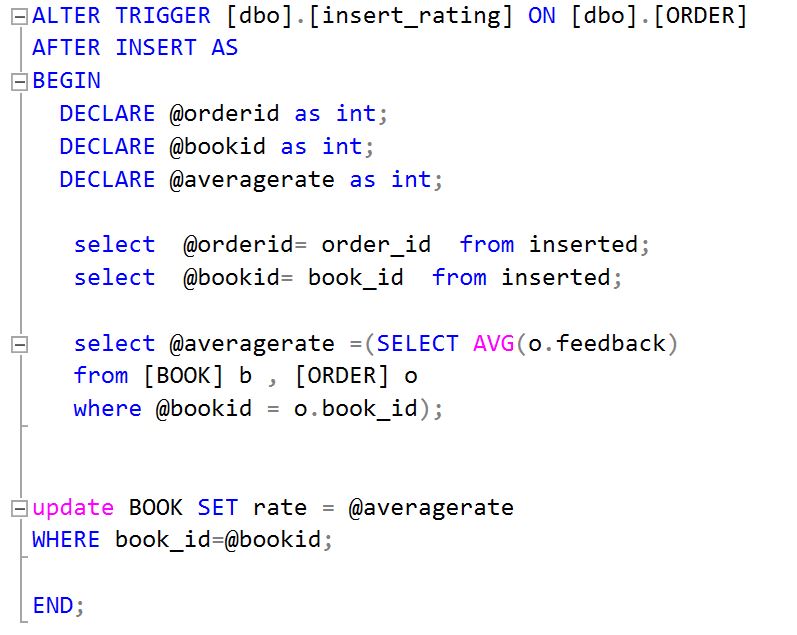
**Triggers**

Increase Sold Quantity Trigger



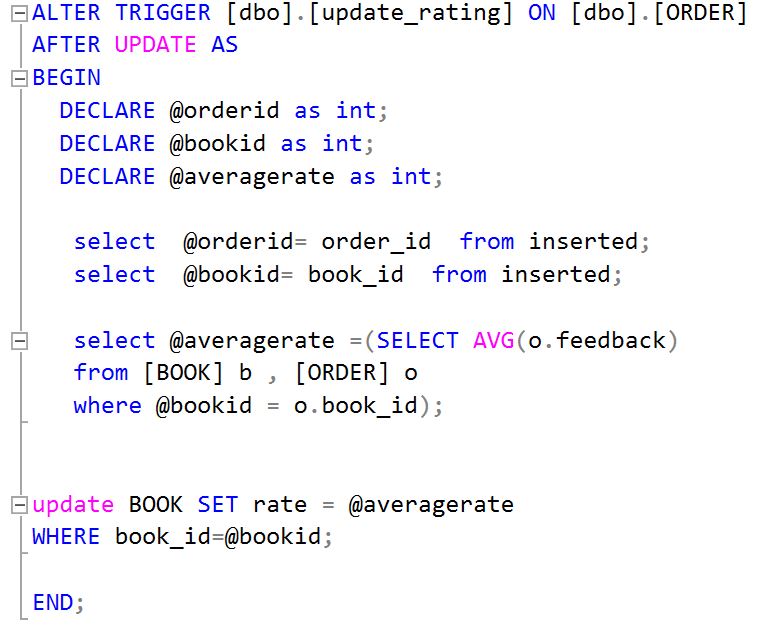
We have used this trigger to increase the sold quantity of a book by one every time the corresponding book is sold.

Insert Rating Trigger



We have created this trigger on order table. It works as follows. Every time a new order is created the users feedback is calculated with all other orders of that book and the rate of that corresponding book changes. Every time a new voting is catched the rate changes (average of all votes). Votes are stored in the feedback variable of order.

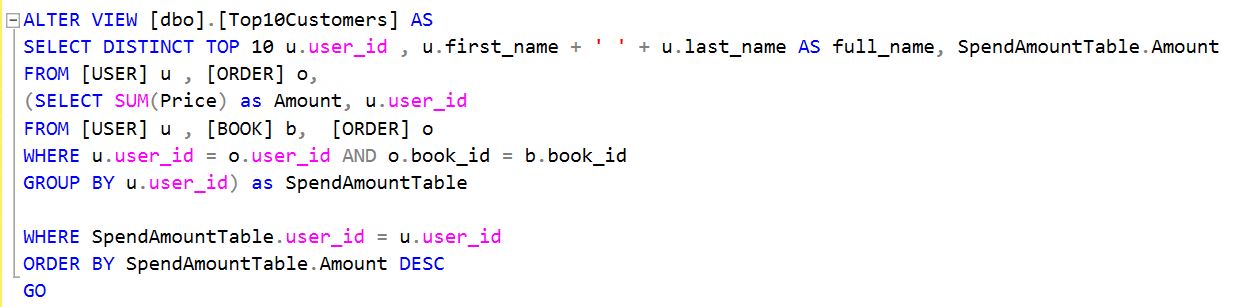
Update Rating Trigger



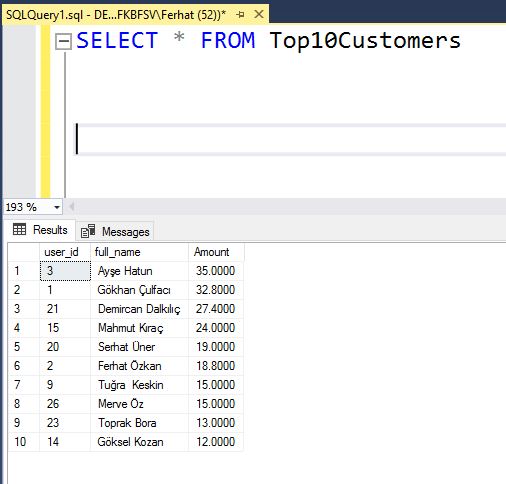
This trigger is almost same as the insert trigger but it is triggered every time the user changes his/her mind and updates his/her vote.

**Views**

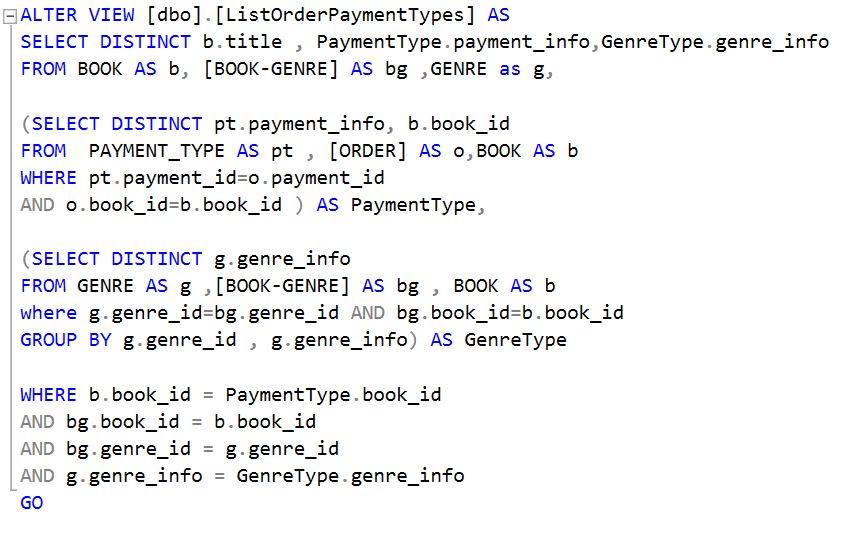
Top10Customers View



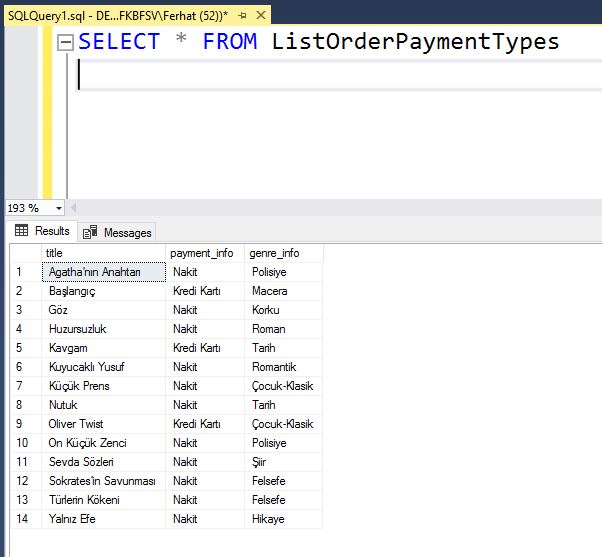
In this view we displayed the users that have spend the most money as top 10. We have combined first name and last name as full name. We have created a inner select query which returns the SpendAmountTable. The inner join returns the users with their total amount that they have spend in the system. From these rows we have used the top 10 with amount descending.



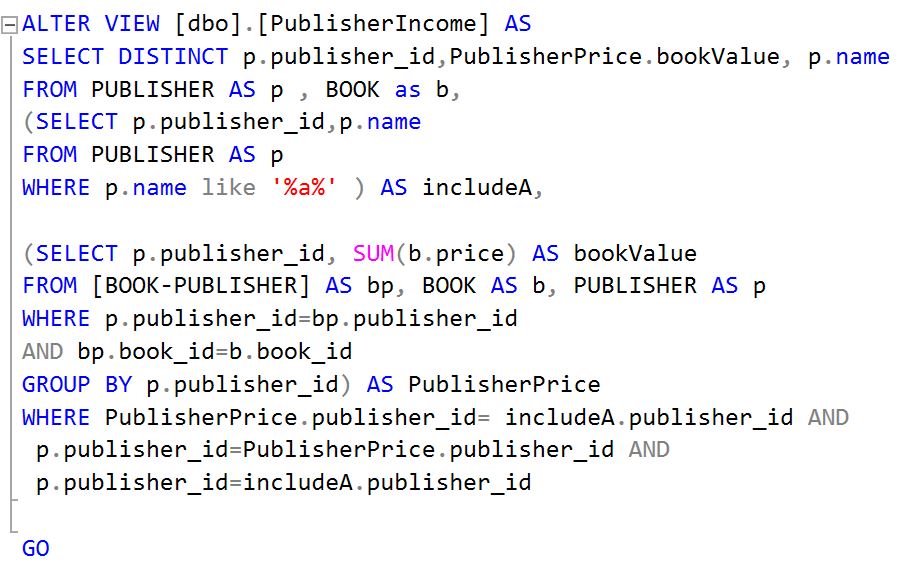
ListOrderPaymentTypes View



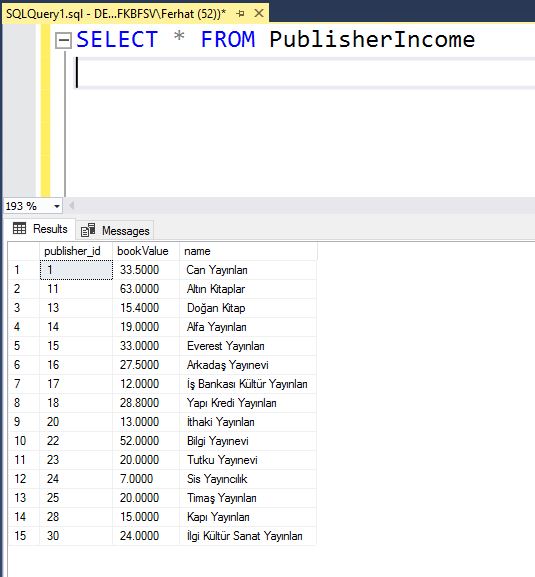
In this view we have returned the genre info of ordered books and the payment type of these orders. We have used two inner select queries one for payment type one for genre type. In the main query we combine these two inner queries.

****

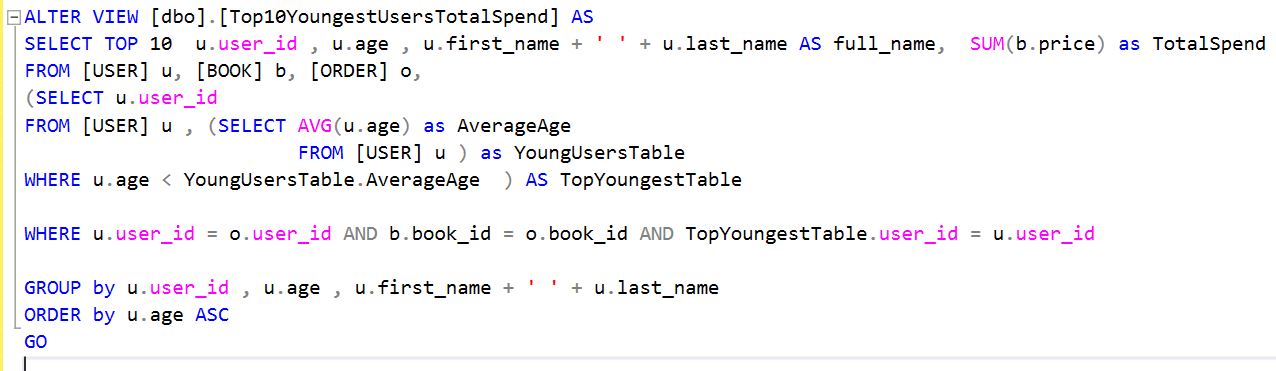
PublisherIncome View



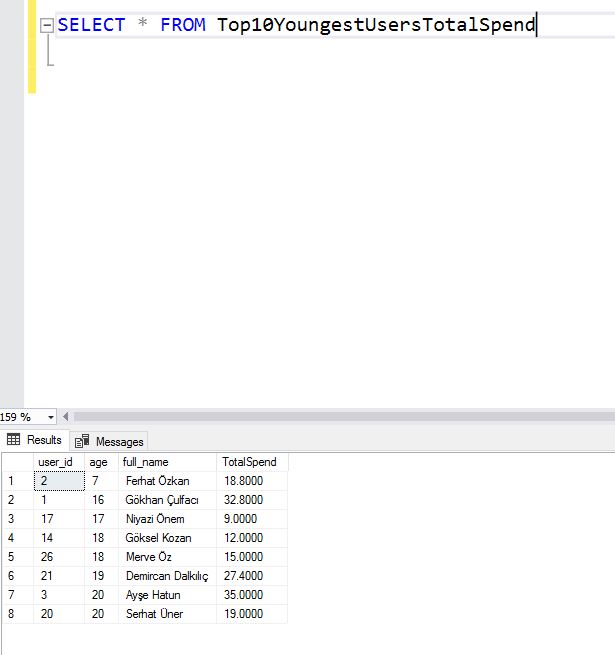
In this view we have displayed the sum of all books prices that the corresponding publisher with a name containing the letter “a” publishes as bookValue and extra informations about the publisher such as publisher id and publisher name. In the inner query PublisherPrice we have calculated the total sum of all books prices by SUM function. And the other inner query includeA returns the publishers with a name containing the letter “a”.

****

Top10YoungestUsersTotalSpend View

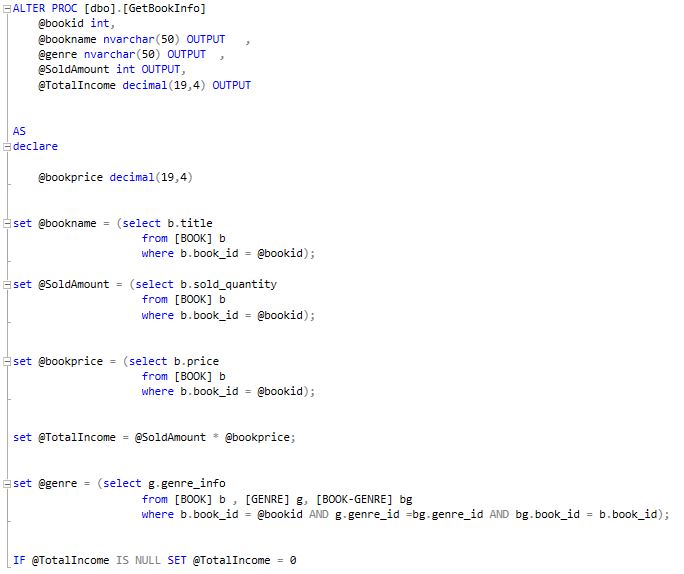


In this view we have displayed the users that are younger than the average of all users and their total sum of all books prices that they have ordered. This is not the total spend money of the user it just the sum of all ordered books different unit prices. (we wanted to display something different than total spend money because we have calculated that already in a procedure). Users are ordered in age ascending order. Extra informations about the user is displayed also such as user id and age.



**Stored Procedures**

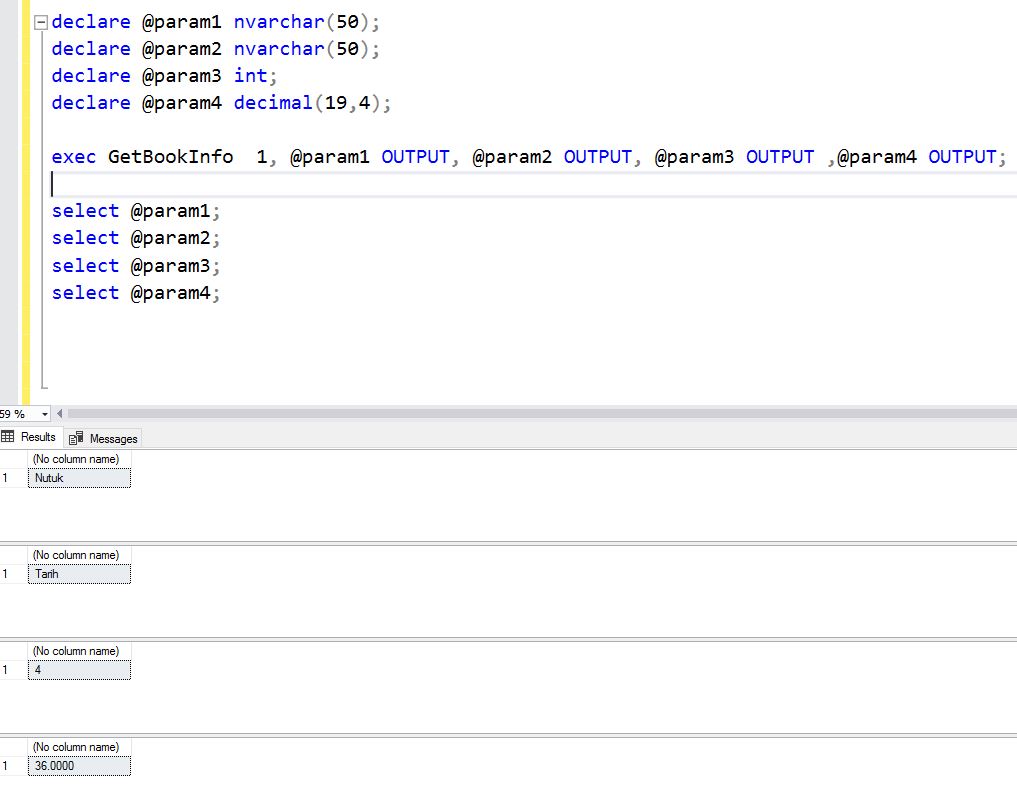
GetBookInfo Procedure



This procedure simply returns informations about the book that is searched with bookid. The ouputs are bookname, genre, SoldAmount and totalincome.

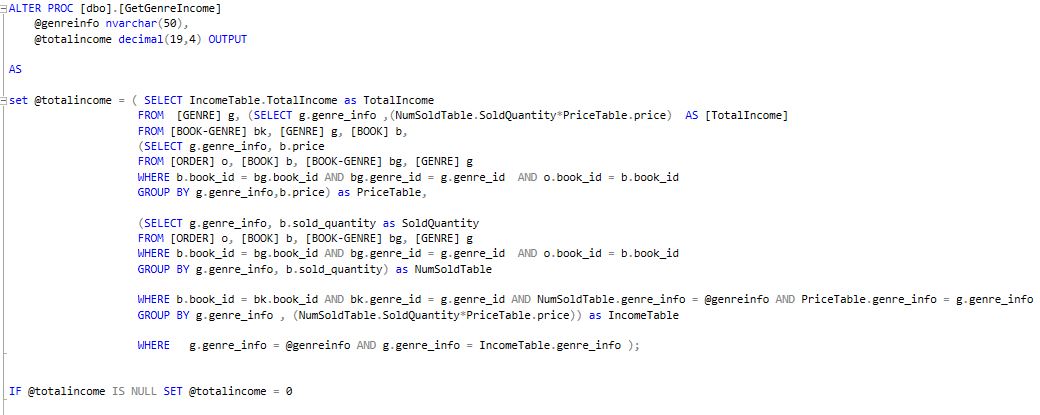
Total income is calculated with soldQuantity\*unitprice.

If there is no order of the book id given from the user the totalincome value will be set to 0.



GetGenreIncome Procedure

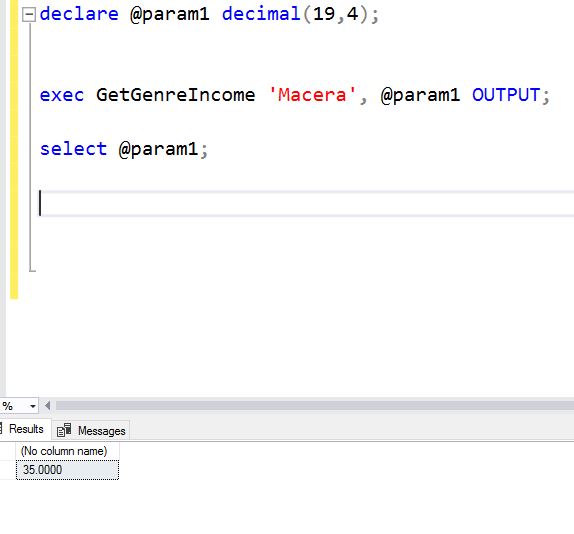
This procedure takes the genre name as input and gives its total income as output.



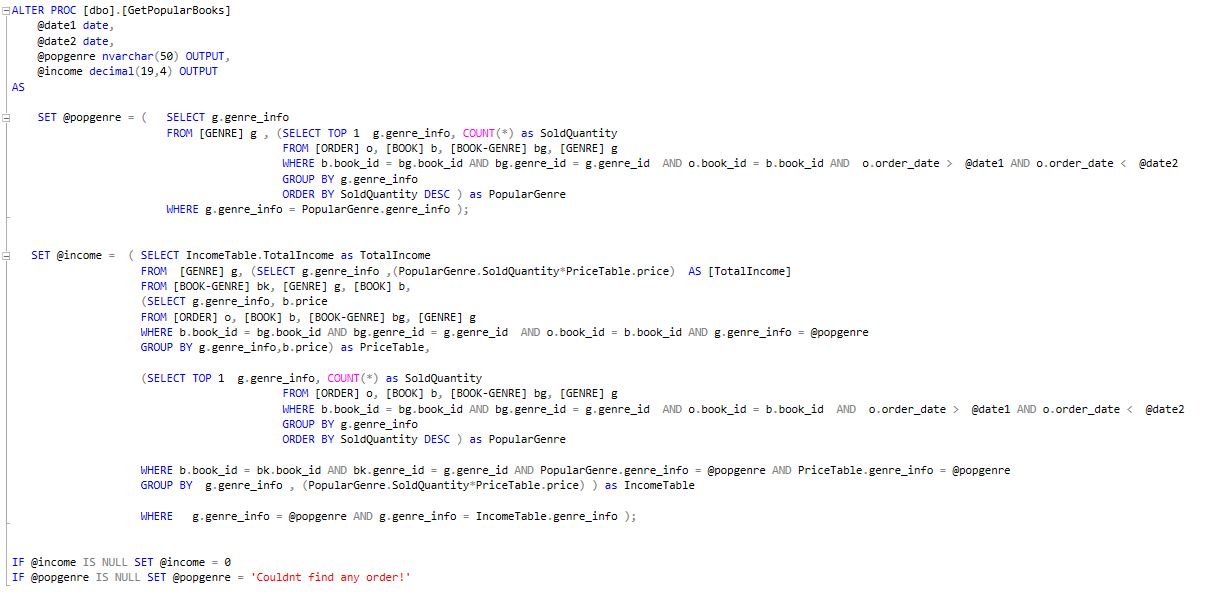
The first inner select query PriceTable returns the books genre info and their prices.

The second inner select query NumSoldTable return the soldquantity attiribute of the corresponding book.

The main select query does the calculation SoldQuantity\*price which is the total income. By the group by command we group these genre infos by this we also sum all the total incomes which gives us a final sum of all income from the taken genre info.



GetPopularBooks Procedure

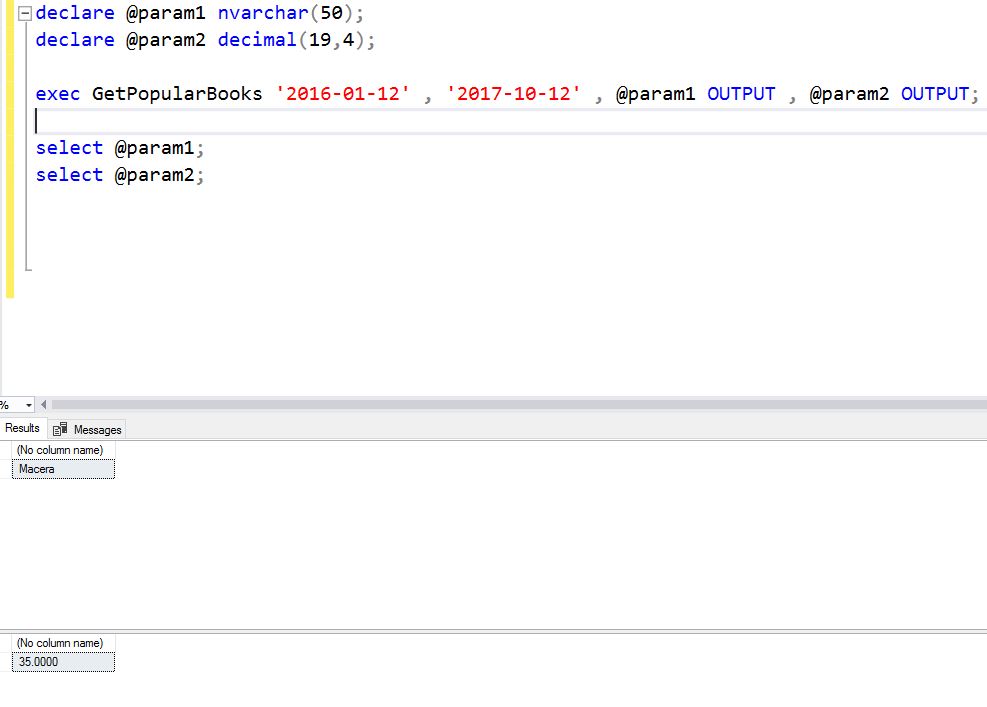


This procedure takes two dates and gives the popgenre which is the genre that sold the most between these dates and the total income from this genre.

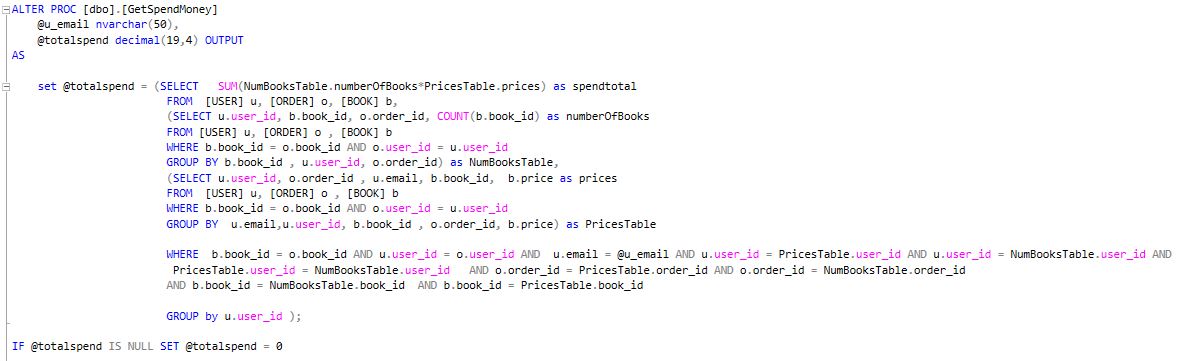
The first query is for calculating the popular genre. For this we wrote a select query that lists all orders between date1 and date2. Then we ordered them with soldQuantity descending and toke the first row. By this we have calculated the order between date1 and date2 with the highest soldQuantity value.

The second query is for calculating the total income of that genre. Which first takes the prices of sold books between these dates and then multiplies the soldQuantity attiributes of these books. And finally adds up all sub totals.

If income is NULL which means no books of that genre was sold the income returns 0 also pop genre returns ‘Couldnt find any order!’.



GetSpendMoney Procedure



This procedure takes user email as input and returns totalspend value which is the total amount that user has spend in the system.

Totalspend is calculated with a select query that has two inner select queries. The first query returns the numofbooks that user has ordered. The second query lists the prices of these books. The main query multiplies these two lists and sums up the subtotal and finally shows us the total amout the user has spend in the system.

