```sql
---1. Процедура перераспределения книг по каталогам:
GO
CREATE PROCEDURE BookRedistribution
   @CatalogName NVARCHAR(50),
   @Keyword NVARCHAR(50),
   @BooksDistributed INT OUTPUT
AS
BEGIN
   SET NOCOUNT ON;

   ---Обновляем записи о книгах с соответствующим ключевым словом и
перераспределяем их по каталогам
   UPDATE Books
   SET Genre = (
       SELECT CatalogName
       FROM Catalogs
       WHERE CatalogName = @CatalogName
   )
   WHERE BookID IN (
       SELECT BookID
       FROM Books
       WHERE Title LIKE '%' + @Keyword + '%'
   );

   ---Получаем количество распределенных книг
   SET @BooksDistributed = @@ROWCOUNT;
END;
GO

---DROP PROCEDURE BookRedistribution

---Пример применения процедуры
DECLARE @BooksDistributed INT;
EXEC BookRedistribution @CatalogName = N'Фантастика', @Keyword = N'Мистика',
@BooksDistributed = @BooksDistributed OUTPUT;
SELECT @BooksDistributed AS 'Books Distributed';
---

---2. Функция для определения рейтинга авторов книг:
USE Leonid_bintsarouski_2
GO
CREATE FUNCTION CalculateAuthorRating
   (@AuthorName NVARCHAR(30))
RETURNS DECIMAL(10, 2)
AS
```

```
BEGIN
   DECLARE @PublicationCount INT;
   DECLARE @ReprintCount INT;
   DECLARE @AuthorRating DECIMAL(10, 2);

   ---Получаем количество публикаций автора
   SELECT @PublicationCount = COUNT(*)
   FROM Books
   WHERE Author = @AuthorName;

   ---Получаем количество переизданий книг автора
   SELECT @ReprintCount = COUNT(*)
   FROM (
       SELECT Title, COUNT(*) AS ReprintCount
       FROM Books
       WHERE Author = @AuthorName
       GROUP BY Title
       HAVING COUNT(*) > 1
   ) AS Reprints;

   ---Вычисляем рейтинг автора
   IF @ReprintCount = 0
       SET @AuthorRating = 0;
   ELSE
       SET @AuthorRating = CAST(@PublicationCount AS DECIMAL(10, 2)) /
@ReprintCount;

   RETURN @AuthorRating;
END;
GO

---DROP FUNCTION CalculateAuthorRating

---использование функции для определения рейтинга авторов книг:
USE Leonid_bintsarouski_2
DECLARE @AuthorName NVARCHAR(30) = N'Agatha Christie';
DECLARE @AuthorRating DECIMAL(10, 2);
SET @AuthorRating = dbo.CalculateAuthorRating(@AuthorName);
SELECT @AuthorRating AS N'Author Rating';
---

---3. Триггер для проверки корректности даты возврата в случае передачи книг во
временное пользование:
USE Leonid_bintsarouski_2;
GO
```

```
CREATE TRIGGER CheckReturnDate
ON BorrowedBooks
AFTER INSERT
AS
BEGIN
   DECLARE @BorrowID INT;
   DECLARE @BorrowDate DATE;
   DECLARE @DueDate DATE;
   DECLARE @ReturnDate DATE;

   SELECT @BorrowID = BorrowID,
          @BorrowDate = BorrowDate,
          @DueDate = DueDate,
          @ReturnDate = ReturnDate
   FROM inserted;

   IF @ReturnDate IS NOT NULL AND @ReturnDate > @DueDate
   BEGIN
       PRINT N'Книга была поздно вернута'
       ---В случае превышения срока возврата вставляем информацию в таблицу Archive
       INSERT INTO Archive (Archive_BookID, Title, Author, YearPublished, Borrower,
ExceededDays)
       SELECT @BorrowID, Bo.Title, Bo.Author, Bo.YearPublished, B.Borrower,
DATEDIFF(DAY, @DueDate, @ReturnDate)
       FROM BorrowedBooks AS B
       JOIN Books AS Bo ON B.BookID = Bo.BookID
       WHERE B.BorrowID = @BorrowID;
   END;
END;
GO

---DROP TRIGGER CheckReturnDate

---Предположим, что вставляем запись о займе книги
INSERT INTO BorrowedBooks (BookID, Borrower, BorrowDate, DueDate, ReturnDate)
VALUES (2, 'John Doe', '2023-05-01', '2023-06-01', '2023-06-02');
DELETE TOP(1) FROM BorrowedBooks
```