

**Федеральное государственное автономное образовательное учреждение высшего  
образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ  
ШКОЛА ЭКОНОМИКИ»**

По направлению подготовки 38.03.05 «Бизнес-информатика»

Образовательная программа «Бизнес-информатика»

Контрольное домашнее задание на тему:

**«Проектирование базы данных для библиотеки»**

Выполнили:

Иксанова Карина Рустамовна, ББИ224

Нелепова Дарья Алексеевна, ББИ224

Палагашвили Диана Михайловна, ББИ226

Фадееенкова Екатерина Дмитриевна, ББИ224

Фролова Юлия Евгеньевна, ББИ224

Москва, 2024

## Оглавление

<b>Оглавление</b>	<b>2</b>
<b>1. Постановка задачи</b>	<b>3</b>
1.1. Анализ предметной области	3
1.2. Инфологическая модель предметной области (ER-диаграмма)	4
<b>2. Даталогическая модель</b>	<b>5</b>
2.1 Описание сущностей	5
2.2. Описание связей	10
2.3. Изображение даталогической модели	12
<b>3. Размещение базы данных в СУБД</b>	<b>13</b>
3.1. Диаграмма БД	13
3.2. Соответствие базы данных третьей нормальной форме (3НФ)	13
4.1. Описание сценариев	14
Сценарий 1: Регистрация новых книг	14
Сценарий 2: Регистрация новых читателей	14
Сценарий 3: Выдача книги читателю	14
Сценарий 4: Возврат книги в библиотеку	15
Сценарий 5: Отчетность по текущему наличию книг	15
Сценарий 6: Отзывы на книги	15
<b>5. Разработка базы данных</b>	<b>16</b>
5.1. Процедуры	16
5.2 Триггеры	20
5.3 Функции	21
5.4 Представления	22
5.5 Запросы	23
5.6 Настройка индексов	28
<b>6. Отчеты в Power BI и Excel.</b>	<b>30</b>
6.1 Основной отчет в Power BI	31
6.2 Детальный отчет в Power BI для drill down	32
6.3 Информационная панель в Excel с графиком и "срезам"	34
Результаты анализа	34
<b>7. Вывод</b>	<b>35</b>
<b>8. Описание вклада участников в проект</b>	<b>36</b>
<b>9. Приложения</b>	<b>37</b>
Скрипт для создания базы данных в MS SQL Server.	37

## **1. Постановка задачи**

### **1.1. Анализ предметной области**

#### Общие сведения

Библиотека – это культурное учреждение, организующее сбор, хранение и общественное пользование книгами. Библиотеки занимаются сбором, хранением, учетом и выдачей читателям литературных произведений, а также информационно-библиографической работой, являются общедоступным источником знаний и основной базой для самообразования. Правильная организация фонда обеспечивает читателю комфортное пользование библиотекой, библиотекарю - быстрое выполнение обязанностей, а также обеспечивает сохранность фондов как общественной собственности. Мы решили создать базу данных, которая будет полезна и удобна для обеих сторон пользования библиотекой.

#### Определение целей и задач

Основная цель использования базы данных библиотеки – это ведение учета книг и управление их выдачей и возвратом. Для этого необходимо:

- Хранить информацию о книгах.
- Хранить информацию о пользователях библиотеки.
- Вести учет выдачи и возврата книг.
- Отслеживать статус книг (в наличии, на руках, утеряна и т.д.).
- Поддерживать отчетность и аналитику.

Дополнительная функциональность: Поддержка системы штрафов при нарушении срока сдачи книги.

- При регистрации (получении читательского билета) в библиотеке читатель вносит депозит, который фиксируется в базе данных. Рекомендуемая сумма — 300 рублей.
- При нарушении срока сдачи книги со счета пользователя ежедневно списывается установленная сумма. Рекомендуемый штраф — 1 руб за каждый день вне срока возврата.
- Читатель не может взять новую книгу, пока не установит сумму на счету равной первоначальному депозиту.
- При удалении читателя из базы данных остаток на счету не возвращается: остаток идет на расходы библиотеки.

Потенциальные пользователи базы данных:

- Администрация библиотеки - могут проводить общее управление базой данных, настраивать права доступа для остальных пользователей, а также добавлять, изменять, редактировать информацию во всех таблицах, создавать отчеты и проводить аналитику.
- Сотрудники библиотеки - могут добавлять, изменять, редактировать информацию во всех таблицах базы данных, создавать отчеты.

- Читатели - имеют доступ к своей личной информации в базе данных, а также информации о книгах, содержащихся в библиотеке, и отзывах на них.

## 1.2. Инфологическая модель предметной области (ER-диаграмма)

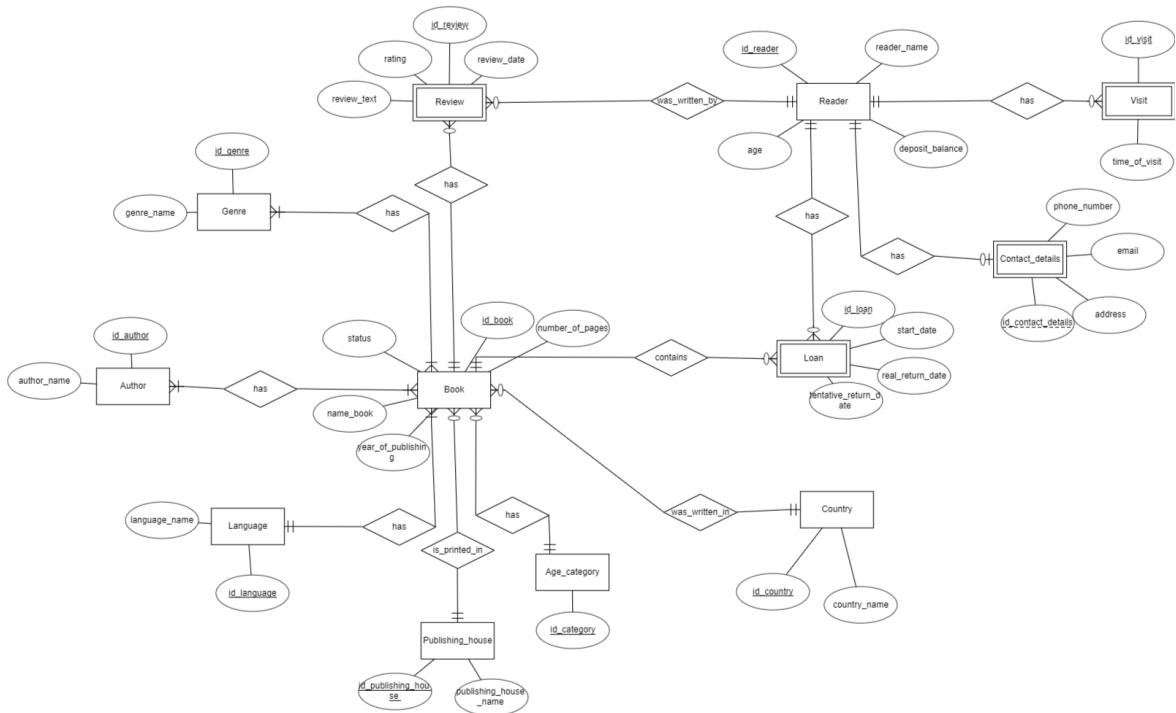


Рис. 1. Схема инфологической модели

## 2. Даталогическая модель

### 2.1 Описание сущностей

#### Таблицы (сущности):

##### 1. Книги

Сущность	Атрибут	Тип данных	Описание	Ключ
Book (сильная)	id_book	int	ID книги	PK
	name_book	nvarchar(150)	Название книги	
	year_of_publishing	date	Год издания	
	id_publishing_house	int	ID издательства	FK
	id_country	int	ID страны	FK
	id_language	int	ID языка	FK
	status	bit	Статус: 1 – в наличии в библиотеке 0 – на руках у читателя	
	number_of_pages	int	Количество страниц	
	id_category	varchar(3)	ID возрастной категории	FK

##### 2. Категория книг

Сущность	Атрибут	Тип данных	Описание	Ключ
Age_category (сильная)	id_category	varchar(3)	ID категории	PK

### 3. Читатели

Сущность	Атрибут	Тип данных	Описание	Ключ
Reader (сильная)	id_reader	int	ID читателя	PK
	reader_name	nvarchar(100)	Фио читателя	
	age	int	Возраст читателя	
	deposit_balance	smallmoney	Остаток по депозиту	

### 4. Контакты

Сущность	Атрибут	Тип данных	Описание	Ключ
Contact_details (слабая)	id_contact_details	int	ID контактных данных	PK
	id_reader	int	ID читателя	FK
	phone_number	nvarchar(50)	Номер телефона читателя	
	email	nvarchar(100)	Email читателя	
	address	nvarchar(200)	Адрес читателя	

### 5. Выдача

Сущность	Атрибут	Тип данных	Описание	Ключ
Loan (слабая)	id_loan	int	ID выдачи	PK
	id_book	int	ID книги	FK
	id_reader	int	ID читателя	FK
	start_date	date	Дата начала аренды	

	tentative_return_date	date	Дата планируемой сдачи	
	real_return_date	date	Дата реальной сдачи	

#### 6. Авторы

Сущность	Атрибут	Тип данных	Описание	Ключ
Author (сильная)	id_author	int	ID автора	PK
	author_name	nvarchar(100)	ФИО автора	

#### 7. Жанры

Сущность	Атрибут	Тип данных	Описание	Ключ
Genre (сильная)	id_genre	int	ID жанра	PK
	genre_name	nvarchar(100)	Наименование жанра	

#### 8. Издательства

Сущность	Атрибут	Тип данных	Описание	Ключ
Publishing_house (сильная)	id_publishing_house	int	ID издательства	PK
	publishing_house_name	nvarchar(100)	Название издательства	

#### 9. Отзывы

Сущность	Атрибут	Тип данных	Описание	Ключ
Review (слабая)	id_review	int	ID отзыва	PK

	id_reader	int	ID читателя	FK
	id_book	int	ID книги	FK
	review_date	datetime	Дата написания отзыва	
	rating	int	Оценка	
	review_text	ntext	Текст отзыва	

#### 10. Журнал посещений

Сущность	Атрибут	Тип данных	Описание	Ключ
Visit (слабая)	id_visit	int	ID посещения	PK
	id_reader	int	ID читателя	FK
	time_of_visit	datetime	Время посещения	

#### 11. Языки

Сущность	Атрибут	Тип данных	Описание	Ключ
Language (сильная)	id_language	int	ID языка	PK
	language_name	nvarchar(100)	Наименование языка	

#### 12. Страны

Сущность	Атрибут	Тип данных	Описание	Ключ
Country	id_country	int	ID страны	PK



(сильная)	country_name	nvarchar(100)	Название стран	
-----------	--------------	---------------	----------------	--

Таблицы, представленные ниже, обеспечивают нашей базе данных приведение к 4 нормальной форме. Данные сущности являются перекрестными.

## 2 связующие таблицы:

1. "Book\_has\_author" - связывает таблицы Book и Author

Сущность	Атрибут	Тип данных	Описание	Ключ
Book_has_author	id_author	int	ID автора	PK
	id_book	int	ID книги	PK
	id_author	int	ID автора	FK
	id_book	int	ID книги	FK

2. "Book\_has\_genre" - связывает таблицы Book и Genre

Сущность	Атрибут	Тип данных	Описание	Ключ
Book_has_genre	id_genre	int	ID жанра	PK
	id_book	int	ID книги	PK
	id_genre	int	ID жанра	FK
	id_book	int	ID книги	FK

### Текстовое описание сущностей:

В базе данных были созданы 12 сущностей. Информация во всех сущностях может изменяться с течением времени, например, при поступлении новых книг в библиотеку, с появлением новых читателей и т.д.

1. “Book” - сущность, содержащая информацию о всех книгах в библиотеке.
2. “Age\_category” - сущность, содержащая информацию о возможных возрастных ограничениях для книг.
3. “Reader” - сущность, содержащая информацию о всех читателях данной библиотеки.
4. “Contact\_details” - сущность, содержащая информацию о контактных данных читателей, зарегистрированных в библиотеке
5. “Loan” - сущность, содержащая информацию о выдаче книг читателям.
6. “Author” - сущность, содержащая информацию об авторах доступных для библиотеки книг.
7. “Genre” - сущность, содержащая информацию о жанрах книг.
8. “Language” - сущность, содержащая информацию о языках, на которых могут быть представлены книги в библиотеке.
9. “Country” - сущность, содержащая информацию о странах, где были написаны оригиналы книг, которые могут быть представлены в библиотеке.
10. “Publishing\_house” - сущность, содержащая информацию об издательствах, которые выпустили книги, которые могут содержаться в библиотеке.
11. “Review” - сущность, содержащая информацию об оставленных читателями отзывах на книги.
12. “Visit” - сущность, содержащая информацию о посещениях библиотеки читателями.

### 2.2. Описание связей

Все представленные связи являются неидентифицирующими.

№	Связь	Отношение	Главная сущность	Подчиненная сущность
1	Book - Author	Многие ко многим	Author	Book
2	Book - Language	Один ко многим	Language	Book
3	Book - Publishing_house	Один ко многим	Publishing_house	Book
4	Book - Country	Один ко многим	Country	Book

5	Book - Loan	Один ко многим	Book	Loan
6	Book - Genre	Многие ко многим	Genre	Book
7	Book - Review	Один ко многим	Book	Review
8	Book - age_category	Один ко многим	Age_category	Book
9	Reader - Loan	Один ко многим	Reader	Loan
10	Reader - Contact_details	Один к одному	Reader	Contact_details
11	Reader - Visit	Один ко многим	Reader	Visit

#### Текстовое описание связей

1. “Book - Author” - определяет, каким автором написана данная книга. “Многие ко многим” - один автор может написать много книг и одна книга может быть написана несколькими авторами. Связь “Book - Author” является обязательной (у книги обязательно есть не менее одного автора).
2. “Book - Language” - информирует о том, на каком языке представлена данная книга. “Один ко многим” - книга может быть написана только на одном языке, а на одном языке написано много книг. Связь “Book - Language” обязательна (книга обязательно представлена на каком-либо языке).
3. “Book - Publishing\_house” - информирует о том, из какого издательства данная книга. “Один ко многим” - определенная книга (под собственным id) может быть выпущена только одним издательством, одно и то же издательство может выпустить много книг. Связь “Book - Publishing\_house” обязательна, так как книга выпускается из определенного издательства.
4. “Book - Country” - информирует о том, в какой стране была написана данная книга. “Один ко многим” - книга может быть написана только в одной стране, в одной стране написано много книг. Связь “Book - Country” обязательна, так как книга должна быть написана в той или иной стране.
5. “Book - Loan” - данная связь обеспечивает накопление записей о том, какая книга была выдана/возвращена в библиотеку. “Один ко многим” - у книги может быть много выдач, но одна запись о выдаче свидетельствует о выдаче только одной книги. Связь “Book - Loan” обязательна, так как выдача не может быть записана без информации о книге.

6. “Book - Genre” - информирует о том, какого жанра данная книга. “Многие ко многим” - у одной книги может быть несколько жанров, в библиотеке может быть зарегистрирована много книг одного жанра. Связь “Book - Genre” обязательна - книга так или иначе имеет какой-либо жанр.
7. “Book - Review” - обеспечивает накопление отзывов на определенную книгу. “Один ко многим” - отзыв может быть создан только на одну книгу, книга может иметь много отзывов. Связь “Book - Review” обязательна - отзыв создается только на определенную книгу.
8. “Book - Age\_category” - информирует о возрастной категории книги. “Один ко многим” - у книги может быть только одна возрастная категория, но у одной возрастной категории может быть множество книг. Связь “Book - Age\_category” обязательна, так как у книги обязательно есть возрастное ограничение.
9. “Reader - Loan” - данная связь обеспечивает накопление записей о том, какой читатель получил ту или иную книгу. “Один ко многим” - разные книги читателю могут быть выданы много раз, запись о выдаче включает в себя запись только об одном читателе. Связь “Reader - Loan” обязательна, так как запись о выдаче подразумевает запись о читателе
10. “Reader - Contact\_details” - связывает читателя и его контактные данные. “Один к одному” - читатель обладает контактными данными в единственном экземпляре, контактные данные могут быть сформированы только на одного читателя. Связь “Reader - Contact\_details” обязательна, так как контактные данные формируются только при условии существования читателя.
11. “Reader - Visit” - данная связь обеспечивает накопление записей о посещениях всех пользователей библиотекой. “Один ко многим” - читатель посещает библиотеку много раз, запись о посещении сообщает только об одном читателе. Связь “Reader - Visit” обязательна - посещение подразумевает информирование о читателе.

### 2.3. Изображение даталогической модели

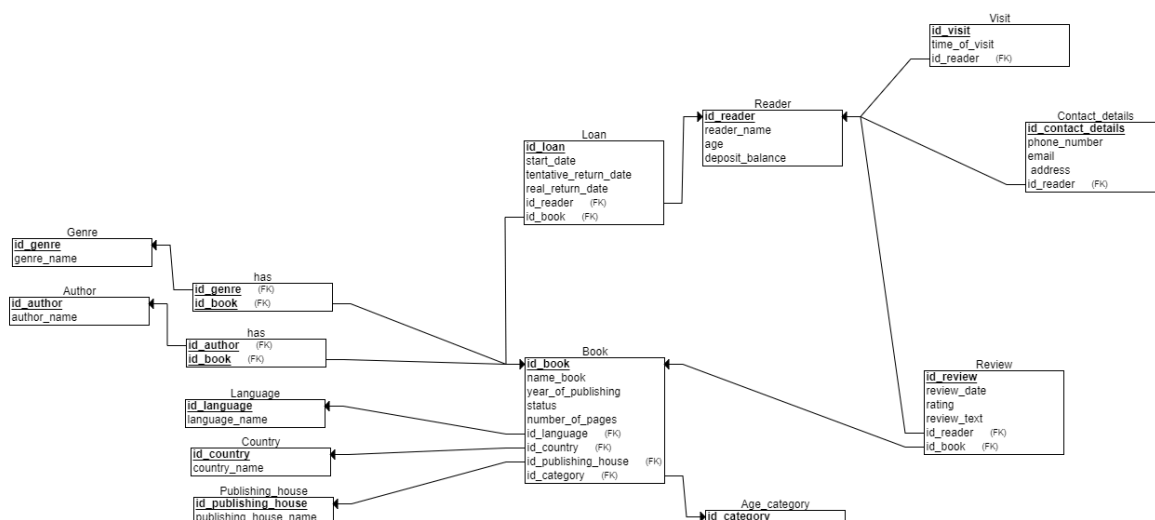


Рис. 2. Схема даталогической модели

### 3. Размещение базы данных в СУБД

#### 3.1. Диаграмма БД

С помощью SQL - кода мы создали все представленные выше сущности и связи. По завершении СУБД сгенерировала диаграмму, изображенную ниже. Базу данных мы заполнили тестовыми данными.

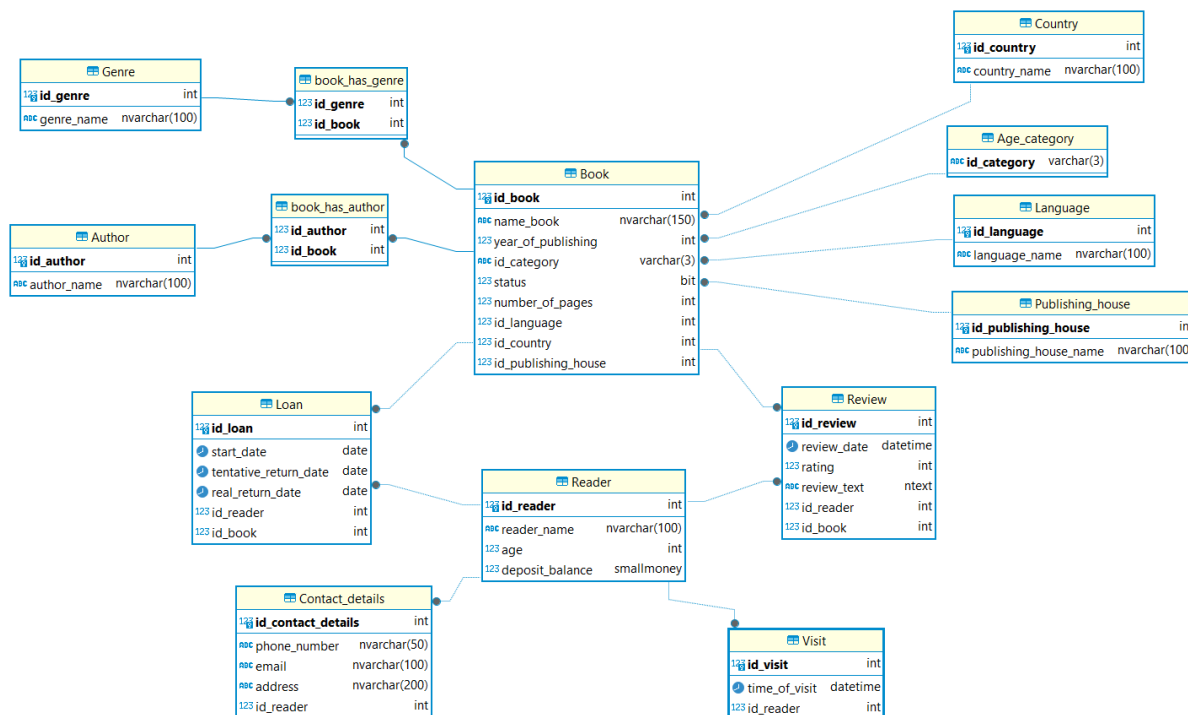


Рис. 3. Диаграмма базы данных.

#### 3.2. Соответствие базы данных третьей нормальной форме (3НФ)

База данных bib150 соответствует третьей нормальной форме (3НФ), так как все таблицы:

- Имеют атомарные значения (1НФ).
- Имеют полную зависимость неключевых атрибутов от первичных ключей (2НФ).
- Не имеют транзитивных зависимостей (3НФ).

Это означает, что структура базы данных нормализована и должна эффективно предотвращать избыточность и аномалии при обновлении данных.

## 4.1. Описание сценариев

### Сценарий 1: Регистрация новых книг

Библиотекарь получает новые книги.

Вводит информацию о книге в систему:

- Название
- Автор(ы) (если автор уже существует в базе, выбирается из списка)
- Жанр (выбирается из списка или добавляется новый)
- Год издания
- Издательство (выбирается из списка или добавляется новое)
- Категория | возрастное ограничение
- Язык (выбирается из списка или добавляется новый)
- Страна происхождения (выбирается из списка или добавляется новая)
- Количество страниц

Информация сохраняется в таблице Book.

### Сценарий 2: Регистрация новых читателей

Новый пользователь приходит в библиотеку и заполняет регистрационную форму.

Библиотекарь вводит данные читателя в систему:

Данные будут введены в двух таблицах (Reader и Contact\_details):

В таблице Reader:

- ФИО
- Возраст
- Размер депозита (по умолчанию - 300 рублей)

В таблице Contact\_details:

- Уникальный код читателя
- Номер телефона
- Email
- Адрес

### Сценарий 3: Выдача книги читателю

Читатель выбирает книгу и подходит к библиотекарю.

Библиотекарь находит книгу в системе и проверяет её статус.

Если книга доступна, библиотекарь создает новую запись в таблице Loan:

- ID книги
- ID читателя
- Дата выдачи книги
- Дата планируемой сдачи

Статус книги меняется на “Не в наличии” (0)

#### **Сценарий 4: Возврат книги в библиотеку**

Читатель возвращает книгу.

Библиотекарь находит соответствующую запись в таблице Loan и обновляет её:

- Дата реальной сдачи

Статус книги обновляется на "в наличии" (1).

Если есть просрочка, библиотекарь добавляет штраф читателю в поле наличие штрафов.

#### **Сценарий 5: Отчетность по текущему наличию книг**

Библиотекарь запрашивает отчет о текущем наличии книг.

Система выбирает все книги из таблицы Book со статусом "в наличии".

Формируется отчет, отображающий:

- Название книги
- Автор
- Жанр
- Количество экземпляров

#### **Сценарий 6: Отзывы на книги**

Читатель может оставить отзыв на книгу.

В системе создается новая запись в таблице Review:

- ID книги
- ID читателя
- Дата
- Текст
- Оценка

Отзыв отображается в карточке книги.

## 5. Разработка базы данных

### 5.1. Процедуры

- 1) База данных должна содержать процедуру, выполняющую обновление на основе результатов другого запроса

Процедура `UpdateBookStatus` выполняет обновление статуса книг в таблице `Book` на основе информации из таблицы `Loan`. Конкретно, она устанавливает статус книг, которые были возвращены (то есть, имеют ненулевое значение `real_return_date`), в 1.

```
CREATE PROCEDURE UpdateBookStatus
AS
BEGIN
    DECLARE @bookId INT;

    DECLARE book_cursor CURSOR FOR
    SELECT id_book
    FROM Loan
    WHERE real_return_date IS NOT NULL;

    OPEN book_cursor;

    FETCH NEXT FROM book_cursor INTO @bookId;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        UPDATE Book
        SET status = 1
        WHERE id_book = @bookId;

        FETCH NEXT FROM book_cursor INTO @bookId;
    END;

    CLOSE book_cursor;
    DEALLOCATE book_cursor;
END;
```



2) Процедура, использующая ветвление и содержащая 3 запроса;

Процедура UpdateReaderDepositBasedOnActivity будет обновлять депозитный баланс читателя на основе его активности. Если у читателя есть активные займы, депозит будет уменьшен на 50. Если у читателя есть отзывы, оставленные в последний месяц, депозит будет увеличен на 100. Если читатель посетил библиотеку за последний месяц, депозит будет увеличен на 30.

```
CREATE PROCEDURE UpdateReaderDepositBasedOnActivity
AS
BEGIN

    DECLARE @currentDate DATE = GETDATE();

    UPDATE Reader
    SET deposit_balance = deposit_balance - 50
    WHERE id_reader IN (
        SELECT DISTINCT id_reader
        FROM Loan
        WHERE real_return_date IS NULL
    );

    UPDATE Reader
    SET deposit_balance = deposit_balance + 100
    WHERE id_reader IN (
        SELECT DISTINCT id_reader
        FROM Review
        WHERE review_date >= DATEADD(MONTH, -1,
@currentDate)
    );

    UPDATE Reader
    SET deposit_balance = deposit_balance + 30
    WHERE id_reader IN (
        SELECT DISTINCT id_reader
        FROM Visit
        WHERE time_of_visit >= DATEADD(MONTH, -1,
@currentDate)
    );

    PRINT 'Обновление депозитного баланса читателей
выполнено успешно.';

END;
```

- 3) Одна из процедур должна включать в себя обработку ошибок TRY CATCH, а также работу с транзакциями (begin transaction, rollback transaction, commit transaction).

Процедура для добавления нового читателя в базу данных. Добавляет нового читателя и его контактные данные в соответствующие таблицы. Использует транзакцию для обеспечения целостности данных.

```
CREATE PROCEDURE AddNewReader
    @reader_name NVARCHAR(100) ,
    @age INT,
    @deposit_balance SMALLMONEY,
    @phone_number NVARCHAR(50) ,
    @email NVARCHAR(100) ,
    @address NVARCHAR(200)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;
        INSERT INTO Reader (reader_name, age,
deposit_balance)
VALUES (@reader_name, @age, @deposit_balance);
        DECLARE @new_reader_id INT;
        SET @new_reader_id = SCOPE_IDENTITY();
        INSERT INTO Contact_details (id_reader,
phone_number, email, address)
VALUES (@new_reader_id, @phone_number, @email,
@address);
        COMMIT TRANSACTION;
        PRINT 'Новый читатель успешно добавлен';
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;
        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity,
@ErrorState);
    END CATCH
END;
```

- 4) Процедуру, которая оформляет выдачу книги читателю. Эта процедура принимает идентификаторы книги и читателя, а также дату предполагаемого возврата книги.

```
CREATE PROCEDURE IssueBookToReader
    @ReaderID INT,
    @BookID INT,
    @TentativeReturnDate DATE
AS
BEGIN
    INSERT INTO Loan (tentative_return_date, id_reader,
id_book)
    VALUES (@TentativeReturnDate, @ReaderID, @BookID);
    INSERT INTO Visit (id_reader)
    VALUES (@ReaderID);
END;
```

- 5) Процедура для возврата книги читателем. Эта процедура принимает идентификаторы книги и читателя, а также дату фактического возврата книги. Процедура также проверяет, если реальная сдача позже планируемой, то за каждый просроченный день вычитаем 10 рублей из депозита читателя.

```
CREATE PROCEDURE ReturnBookFromReader
    @ReaderID INT,
    @BookID INT,
    @RealReturnDate DATE
AS
BEGIN
    DECLARE @TentativeReturnDate DATE;
    DECLARE @DaysLate INT;
    DECLARE @Penalty MONEY;
    SELECT @TentativeReturnDate = tentative_return_date
FROM Loan
    WHERE id_reader = @ReaderID AND id_book = @BookID AND
real_return_date IS NULL;
    UPDATE Loan
    SET real_return_date = @RealReturnDate
    WHERE id_reader = @ReaderID AND id_book = @BookID AND
real_return_date IS NULL;
    IF @RealReturnDate > @TentativeReturnDate
    BEGIN
        SET @DaysLate = DATEDIFF(DAY, @TentativeReturnDate,
@RealReturnDate);
        SET @Penalty = @DaysLate * 10;
        UPDATE Reader
        SET deposit_balance = deposit_balance - @Penalty
    END
```

```

WHERE id_reader = @ReaderID;
END
END;

```

## 5.2 Триггеры

1) **Цель:** Обновить статус книги на "не в наличии" (0) в таблице **Book**, когда создается новая запись в таблице **Loan**.

**Логика работы триггера:**

1. **Событие срабатывания:** Триггер срабатывает после вставки (**INSERT**) новой записи в таблицу **Loan**.
2. **Действия:** После того, как новая запись была вставлена в таблицу **Loan**, триггер выполняет обновление статуса книги в таблице **Book**.
  - Триггер использует внутреннюю таблицу **inserted**, которая содержит новые записи, вставленные в таблицу **Loan**.
  - Для каждой новой записи в **Loan**, триггер находит соответствующую книгу в таблице **Book** (сопоставляя **id\_book** из таблицы **inserted** с **id\_book** в таблице **Book**) и обновляет статус этой книги на 0 (не в наличии).

```

CREATE TRIGGER trg_update_book_status_on_loan
ON Loan
AFTER INSERT
AS
BEGIN
    UPDATE Book
    SET status = 0
    FROM Book b
    JOIN inserted i ON b.id_book = i.id_book;
END;

```

2) **Цель:** Обновить депозит читателя в таблице **Reader**, если книга была возвращена позже планируемой даты возврата (**tentative\_return\_date**), вычитая по 10 рублей за каждый день просрочки.

**Логика работы триггера:**

1. **Событие срабатывания:**
  - Триггер срабатывает после обновления записи в таблице **Loan**.
2. **Действия:**

- После обновления записи в таблице **Loan**, триггер выполняет обновление депозита в таблице **Reader** для тех случаев, когда книга была возвращена позже планируемой даты возврата.
- Триггер использует внутреннюю таблицу **inserted**, которая содержит новые данные записей, которые были обновлены.
- Для каждой обновленной записи в **Loan**, если фактическая дата возврата (**real\_return\_date**) больше планируемой даты возврата (**tentative\_return\_date**), триггер вычисляет разницу в днях между этими датами и умножает её на 10.
- Полученная сумма вычитается из депозита читателя (**deposit\_balance**) в таблице **Reader**.

```
CREATE TRIGGER trg_update_reader_deposit_on_late_return
ON Loan
AFTER UPDATE
AS
BEGIN
    UPDATE Reader
    SET deposit_balance = deposit_balance - DATEDIFF(day,
1.tentative_return_date, 1.real_return_date) * 10
    FROM Reader r
    JOIN Loan l ON r.id_reader = l.id_reader
    JOIN inserted i ON l.id_loan = i.id_loan
    WHERE 1.real_return_date > 1.tentative_return_date;
END;
```

### 5.3 Функции

- 1) База данных должна содержать функцию для подсчёта количества выдач конкретной книги.

Функция **GetLoanCount** принимает идентификатор книги в качестве параметра и возвращает количество раз, которое эта книга была выдана.

```
CREATE FUNCTION dbo.GetLoanCount (@BookId INT)
RETURNS INT
AS
BEGIN
    DECLARE @LoanCount INT;

    SELECT @LoanCount = COUNT(*)
    FROM Loan
    WHERE id_book = @BookId;
    RETURN @LoanCount;
END;
```

- 2) База данных должна содержать функцию для поиска всех издательств, которые выпустили конкретную книгу.

Функция `GetPublishingHousesForBook` принимает идентификатор книги в качестве параметра и возвращает список всех издательств, которые выпустили данную книгу. Функция ищет в таблице `Book` книгу с нужным идентификатором и возвращает идентификаторы и названия издательств, выпустивших эту книгу, из таблицы `Publishing_house`.

```
CREATE FUNCTION dbo.GetPublishingHousesForBook (@BookId INT)
RETURNS TABLE
AS
RETURN
(
    SELECT ph.id_publishing_house, ph.publishing_house_name
    FROM Publishing_house ph
    INNER JOIN Book b ON ph.id_publishing_house =
b.id_publishing_house
    WHERE b.id_book = @BookId
);
```

## 5.4 Представления

- 1) Представление для получения топ-10 книг с возрастным ограничением 12+ или 16+ (подростковые книги) по количеству выдач читателям за последние 6 месяцев.

```
CREATE VIEW Top10TeenBooks AS
SELECT TOP 10 b.name_book,
COUNT(*) AS number_of_loans
FROM Book b
JOIN Loan l ON b.id_book = l.id_book
WHERE b.id_category IN ('12+', '16+')
AND DATEDIFF(month, l.start_date, GETDATE()) <= 6
GROUP BY b.name_book
ORDER BY number_of_loans DESC,
b.name_book
```

- 2) Представление для получения id и имени читателя, прочитавшего наибольшее количество страниц за последний год.

```
CREATE VIEW TopReaderByPages AS
SELECT TOP 1 r.id_reader,
r.reader_name,
SUM(b.number_of_pages) AS total_pages_read
FROM Reader r
```

```

JOIN Loan l ON r.id_reader = l.id_reader
JOIN Book b ON l.id_book = b.id_book
WHERE DATEDIFF(MONTH, l.start_date, GETDATE()) <= 12
GROUP BY r.id_reader,
         r.reader_name
ORDER BY total_pages_read DESC

```

## 5.5 Запросы

Простой запрос с условием и формулами в SELECT:

- 1) Получение количества свободных экземпляров книги (например, "Мастера и Маргариты"):

```

SELECT name_book, COUNT(*) AS free_copies
FROM Book
WHERE status = 1 AND name_book = 'Мастер и Маргарита'
GROUP BY name_book;

```

- 2) Поиск самых юных пользователей библиотеки:

```

SELECT reader_name, age
FROM Reader
WHERE age = (SELECT MIN(age) FROM Reader);

```

Запрос с коррелированным подзапросом в SELECT:

- 1) Список книг с названием и количеством отзывов к каждой книге

```

SELECT b.id_book, b.name_book,
       (SELECT COUNT(*)
        FROM Review r
        WHERE r.id_book = b.id_book) AS review_count
FROM Book b;

```

- 2) список читателей (имя, id) с количеством книг на руках

```

SELECT r.id_reader, r.reader_name,
       (SELECT COUNT(*)
        FROM Loan l
        WHERE l.id_reader = r.id_reader AND l.real_return_date
        IS NULL) AS books_on_hand
FROM Reader r;

```

Запрос с подзапросом в FROM

- 1) Запрос, определяющий читателя, который взял больше всего книг

```

SELECT TOP 1 top_reader.reader_name,
top_reader.total_books_borrowed
FROM (
    SELECT r.reader_name, COUNT(l.id_book) AS
total_books_borrowed
    FROM Reader r
    JOIN Loan l ON r.id_reader = l.id_reader
    GROUP BY r.reader_name
) AS top_reader
ORDER BY top_reader.total_books_borrowed DESC;

```

2) Все книги которые сейчас на руках у читателей

```

SELECT b.name_book, r.reader_name, l.start_date,
l.tentative_return_date
FROM (
    SELECT id_book, id_reader, start_date,
tentative_return_date
    FROM Loan
    WHERE real_return_date IS NULL
) AS l
JOIN Book b ON l.id_book = b.id_book
JOIN Reader r ON l.id_reader = r.id_reader;

```

### Запросы с подзапросом в FROM, агрегированием, группировкой и сортировкой

1) Топ-5 жанров по количеству книг в возрастной категории 16+, представленных в библиотеке

```

SELECT
    TOP 5 g.genre_name,
COUNT(*) AS number_of_books
FROM
    book_has_genre bhg
JOIN Genre g ON bhg.id_genre = g.id_genre
AND bhg.id_book IN
(
    SELECT b.id_book
    FROM Book b
    WHERE b.id_category = '16+')
GROUP BY g.genre_name
ORDER BY COUNT(*) DESC;

```

2) Топ-5 авторов по количеству книг на русском языке, выданных читателям за последние 6 месяцев.

```

SELECT

```



```

        TOP 5 a.author_name,
        COUNT(*) AS number_of_loans
FROM
    book_has_author bha
JOIN Author a ON
    bha.id_author = a.id_author
    AND bha.id_book IN
    (
        SELECT b.id_book
        FROM Book b
        JOIN [Language] l ON l.id_language = b.id_language
        JOIN Loan lo ON lo.id_book = b.id_book
        WHERE l.language_name = 'Русский'
            AND DATEDIFF(MONTH, lo.start_date, GETDATE()) <=
6)
GROUP BY a.author_name
ORDER BY COUNT(*) DESC,
    a.author_name

```

#### Запрос с коррелированным подзапросом в WHERE

- 1) Данный запрос находит всех читателей, которые взяли книги на более чем 7 дней

```

SELECT reader_name, age, deposit_balance
FROM Reader r
WHERE EXISTS (
    SELECT 1
    FROM Loan l
    WHERE l.id_reader = r.id_reader
    AND DATEDIFF(day, l.start_date, l.real_return_date) > 7
);

```

#### Запрос, использующий оконную функцию LAG или LEAD для выполнения сравнения данных в разных периодах:

- 1) Расчет средней разницы в днях между визитами пользователей

```

WITH VisitDifferences AS (
    SELECT id_reader, time_of_visit,
        LAG(time_of_visit) OVER (PARTITION BY id_reader ORDER
BY time_of_visit) AS previous_visit
    FROM Visit
)
SELECT AVG(DATEDIFF(DAY, previous_visit, time_of_visit)) AS
avg_days_between_visits
FROM VisitDifferences
WHERE previous_visit IS NOT NULL;

```

### Запрос с агрегированием и выражением JOIN, включающим не менее 2 таблиц

- 1) Запрос, возвращающий количество страниц, прочитанных каждым пользователем

```
SELECT r.reader_name, SUM(b.number_of_pages) AS
total_pages_read
FROM Reader r
JOIN Loan l ON r.id_reader = l.id_reader
JOIN Book b ON l.id_book = b.id_book
GROUP BY r.reader_name;
```

- 2) Запрос, выдающий рейтинг каждой книги

```
SELECT b.name_book, AVG(r.rating) AS average_rating
FROM Book b
JOIN Review r ON b.id_book = r.id_book
JOIN book_has_author bha ON b.id_book = bha.id_book
GROUP BY b.name_book;
```

- 3) Топ языков, на которых чаще всего берут книги

```
SELECT l.language_name, COUNT(ln.id_book) AS book_count
FROM Language l
JOIN Book b ON l.id_language = b.id_language
JOIN Loan ln ON b.id_book = ln.id_book
GROUP BY l.language_name
ORDER BY book_count DESC;
```

### Запрос с EXISTS

- 1) Читатели, которые написали хотя бы 1 отзыв

```
SELECT r.id_reader, r.reader_name
FROM Reader r
WHERE
    EXISTS (SELECT 1
            FROM Review rv
            WHERE rv.id_reader = r.id_reader);
```

### Запрос, использующий манипуляции с множествами

- 1) Данный запрос находит имена, которые являются и читателями, и авторами.

```
SELECT reader_name AS name
FROM Reader
INTERSECT
SELECT author_name AS name
FROM Author;
```

### Запрос с внешним соединением и проверкой на наличие NULL

- 1) Отзывы, где проставлена оценка, но не написан текст:

```
SELECT r.id_review, r.review_date, r.rating, r.review_text,
       r.id_reader, r.id_book,
       b.name_book, a.author_name
FROM Review r
LEFT JOIN Book b ON r.id_book = b.id_book
LEFT JOIN book_has_author bha ON b.id_book = bha.id_book
LEFT JOIN Author a ON bha.id_author = a.id_author
WHERE r.rating IS NOT NULL AND (r.review_text IS NULL OR
DATALENGTH(r.review_text) = 0);
```

### Запрос с агрегированием и выражением JOIN, включающим не менее 3 таблиц/выражений

- 1) Средняя оценка книг для каждого автора, основываясь на данных о рецензиях

```
SELECT a.author_name, AVG(1.0 * r.rating) AS average_rating
FROM Author a
JOIN book_has_author bha ON a.id_author = bha.id_author
JOIN Book b ON bha.id_book = b.id_book
JOIN Review r ON b.id_book = r.id_book
GROUP BY a.author_name
ORDER BY average_rating DESC;
```

### Запрос с CASE (IF) и агрегированием

- 1) Классификация читателей по активности за последние 6 месяцев на основе количества взятых книг.

```
SELECT
    r.reader_name AS name,
    CASE
        WHEN count(*) >= 6 THEN 'Особо активный'
        WHEN count(*) <= 2 THEN 'Слабо активный'
        ELSE 'Активный'
    END AS Involvement
FROM
    Reader r
JOIN Loan l ON
    l.id_reader = r.id_reader
WHERE
    DATEDIFF(MONTH, l.start_date, GETDATE()) <= 6
GROUP BY
    r.reader_name;
```

### Запрос с HAVING и агрегированием

1) Список читателей, которые брали >1 книги

```
SELECT l.id_reader, r.reader_name, COUNT(l.id_book) AS
book_count
FROM Loan l
JOIN Reader r ON l.id_reader = r.id_reader
GROUP BY l.id_reader, r.reader_name
HAVING COUNT(l.id_book) >= 2;
```

### Запрос SELECT INTO для подготовки выгрузки

1) Перенос всех читателей, просрочивших книгу на 6 месяцев и более (id, имя читателя и название просроченной книги) в новую таблицу.

```
SELECT
    r.id_reader,
    r.reader_name,
    b.name_book as overdue_book
INTO Black_list
FROM Reader r
JOIN Loan l ON l.id_reader = r.id_reader
JOIN Book b ON l.id_book = b.id_book
WHERE real_return_date IS NULL
    AND DATEDIFF(month, l.tentative_return_date, GETDATE())
    >= 6;
```

## 5.6 Настройка индексов

Исходя из имеющихся запросов, как самый высоконагруженный и длинный мы можем выбрать запрос, который позволяет получить количество книг, взятых конкретными читателями, где количество книг больше или равно 2. Самой большой и наиболее важной таблицей является таблица Loan. Таким образом, были созданы индексы, которые охватывают колонки: id\_reader и id\_book. Данные колонки были выбраны в приведенной последовательности по следующим причинам:

- **id\_reader**: Большинство запросов направлены на получение данных о конкретных читателях, поэтому этот столбец выбран первым.
- **id\_book**: Важный столбец для определения количества книг, взятых читателем.

### Индексы:

```
CREATE INDEX IDX_Loan_id_reader ON Loan(id_reader);
CREATE INDEX IDX_Loan_id_book ON Loan(id_book);
CREATE INDEX IDX_Reader_id_reader ON Reader(id_reader);
```

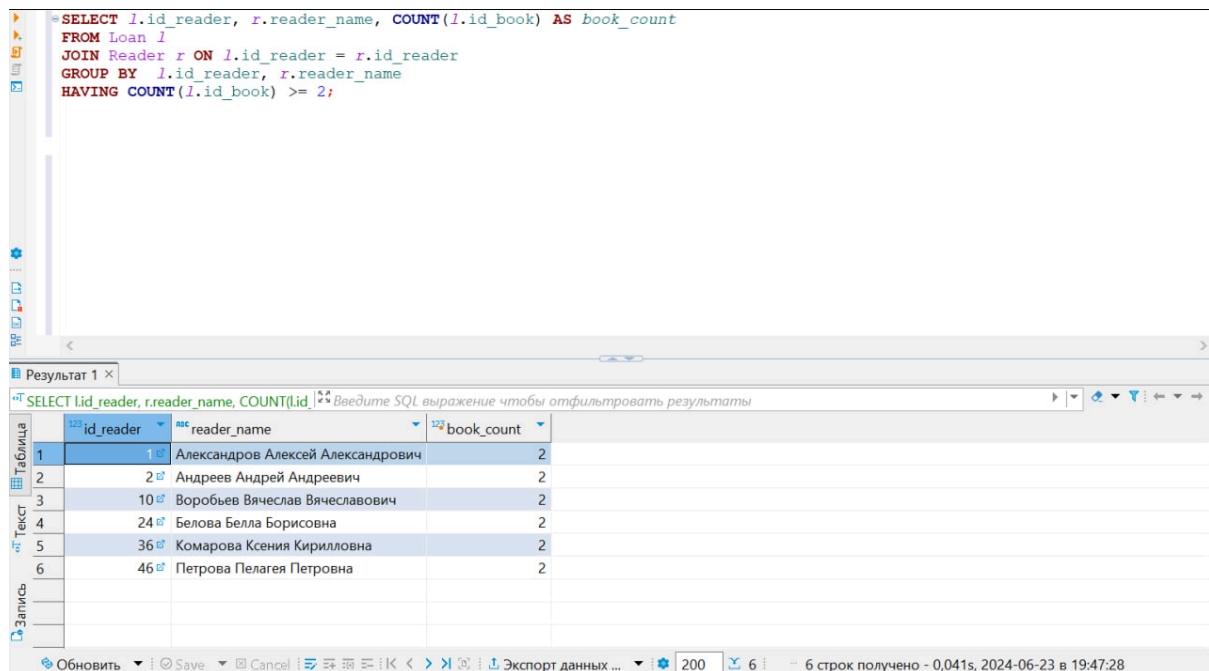
### Запрос:

```

SELECT l.id_reader, r.reader_name, COUNT(l.id_book) AS
book_count
FROM Loan l
JOIN Reader r ON l.id_reader = r.id_reader
GROUP BY l.id_reader, r.reader_name
HAVING COUNT(l.id_book) >= 2;

```

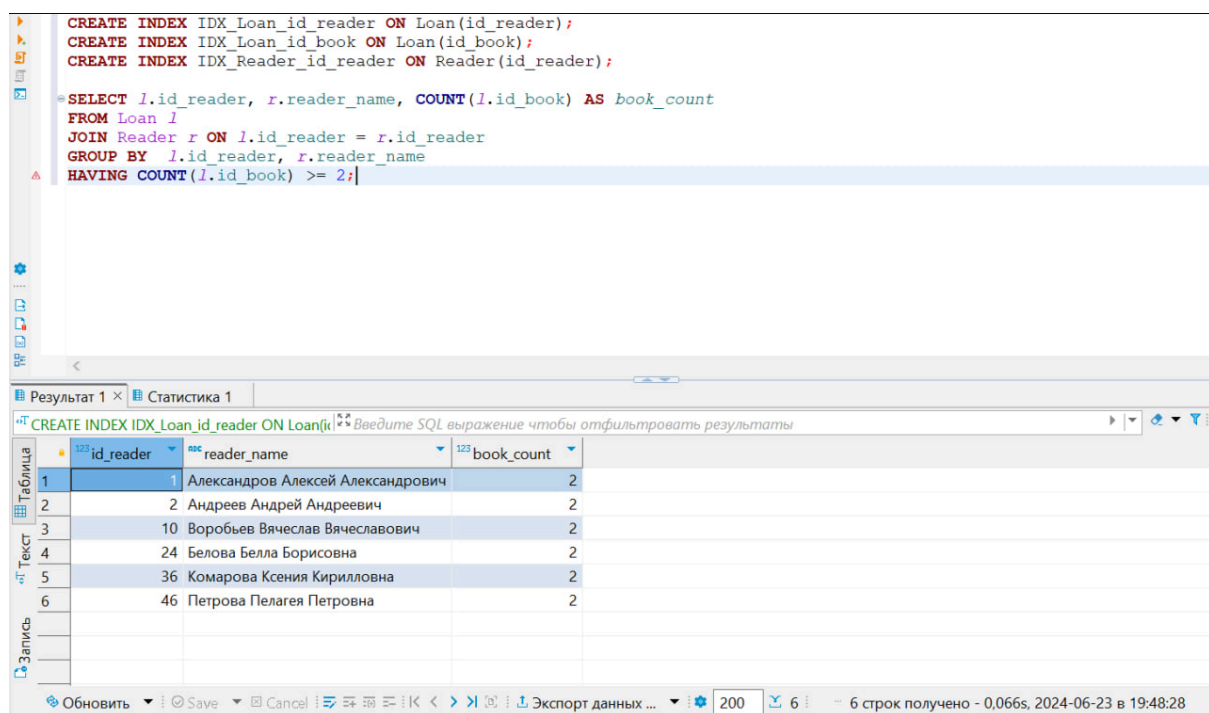
Результаты:



The screenshot shows a database query execution interface. The SQL query is displayed in the top pane, and the results are shown in the bottom pane. The results table has three columns: id\_reader, reader\_name, and book\_count. The data is as follows:

id_reader	reader_name	book_count
1	Александров Алексей Александрович	2
2	Андреев Андрей Андреевич	2
10	Воробьев Вячеслав Вячеславович	2
24	Белова Белла Борисовна	2
36	Комарова Ксения Кирилловна	2
46	Петрова Пелагея Петровна	2

Рис. 4. Время выполнения до применения индекса



The screenshot shows the same database query execution interface as in Figure 4, but with the addition of three indexes: IDX\_Loan\_id\_reader, IDX\_Loan\_id\_book, and IDX\_Reader\_id\_reader. The SQL query is the same, and the results table is identical to the one in Figure 4.

id_reader	reader_name	book_count
1	Александров Алексей Александрович	2
2	Андреев Андрей Андреевич	2
10	Воробьев Вячеслав Вячеславович	2
24	Белова Белла Борисовна	2
36	Комарова Ксения Кирилловна	2
46	Петрова Пелагея Петровна	2

Рис. 5. Время выполнения после применения индекса

На рисунке 4 показано время выполнения запроса до применения индексов: 0.041с. После создания индексов, как показано на рисунке 5, время выполнения запроса увеличилось: 0.066с. Это может быть связано с тем, что таблица содержит малое количество данных, а накладные расходы на использование индексов превысили выгоду от их использования.

**Вывод:**

Как мы можем заметить, созданные индексы не всегда приводят к улучшению времени выполнения запросов, особенно в случае с небольшими таблицами или при неправильном выборе индексов. В данном случае время выполнения запроса увеличилось, что указывает на то, что накладные расходы на использование индексов превысили выгоду от их использования. Поэтому при настройке индексов всегда важно проводить тестирование и анализировать результаты, чтобы убедиться в их эффективности.

## 6. Отчеты в Power BI и Excel.

### 6.1 Основной отчет в Power BI

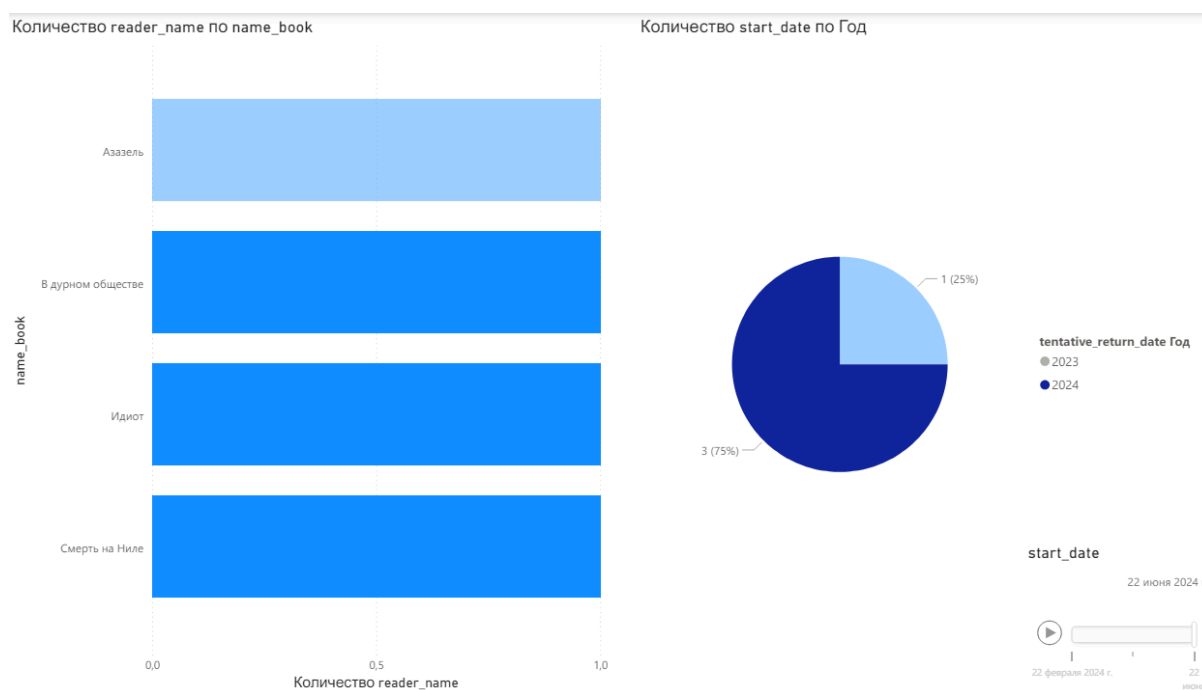


Рис. 6. Отчет в Power BI

В столбчатой диаграмме количество reader\_name по name\_book по оси X записано количество читателей, которые не вернули книгу. По оси Y записаны названия соответствующих книг. Такая диаграмма позволит сотрудникам библиотеки понять, по сколько экземпляров разных книг было утеряно, что поможет заказать необходимые книги для будущего использования.

В круговой диаграмме количество start\_date по Год показано, сколько книг в каком году не было возвращено. Эта диаграмма позволит сотрудникам библиотеки узнать, в каком году было утеряно наибольшее или наименьшее количество книг.

## 6.2 Детальный отчет в Power BI для drill down

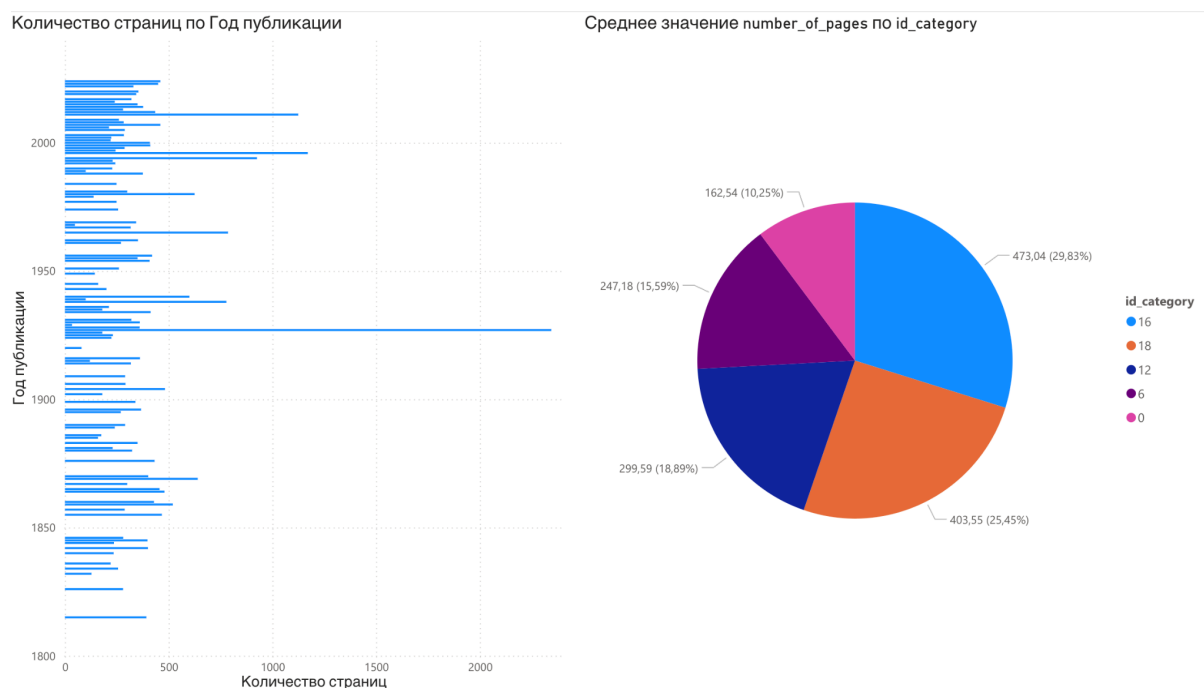


Рис. 7. Отчет в Power BI

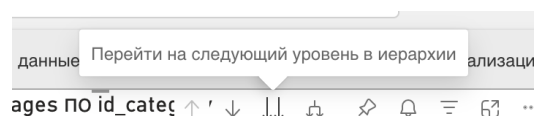
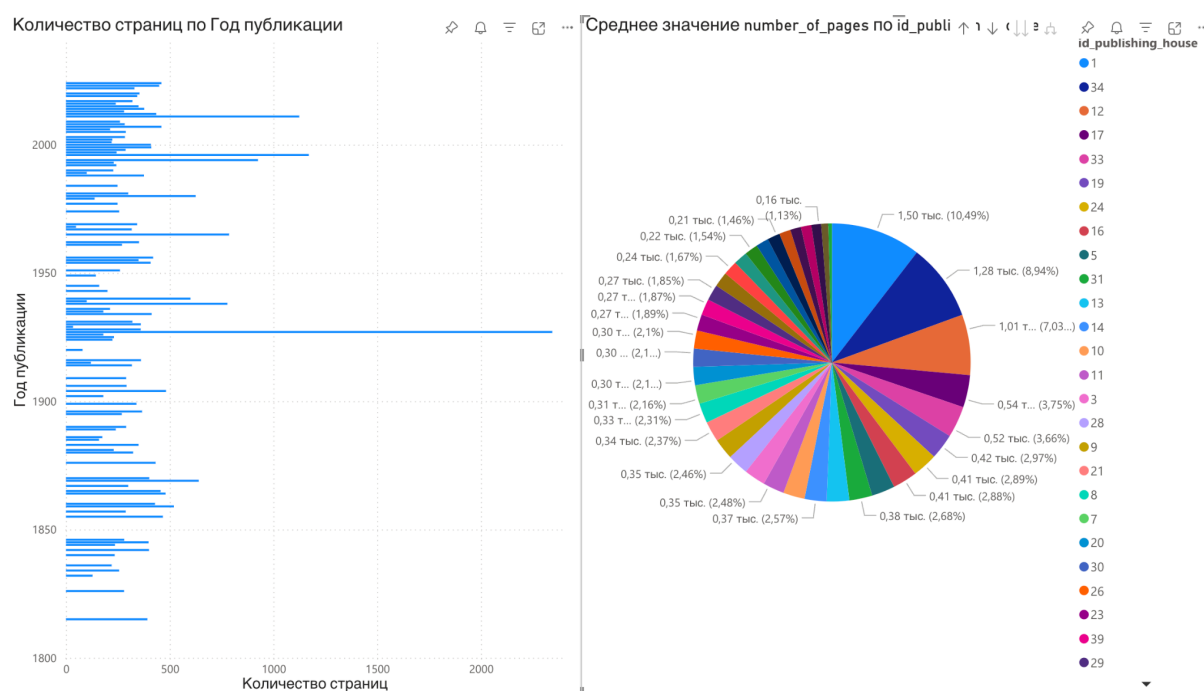


Рис. 8. Отчет в Power BI



*Рис. 9 Отчет в Power BI*



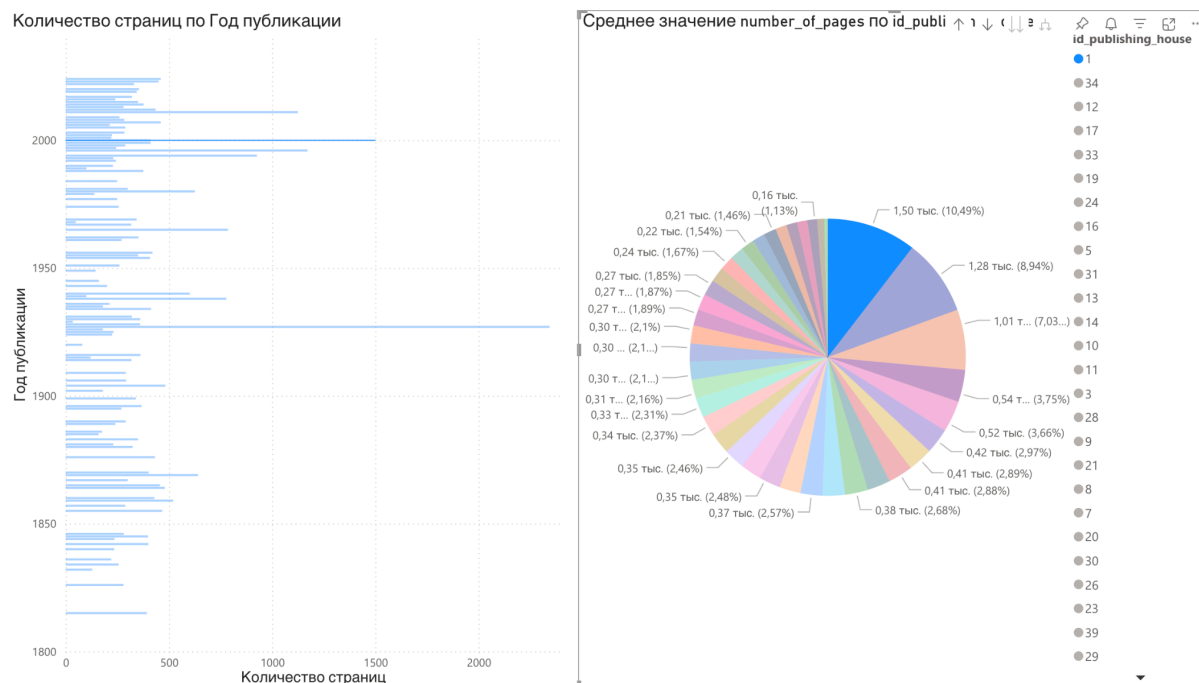


Рис. 10. Отчет в Power BI

На круговой диаграмме (рис. 7) представлено среднее значение страниц по возрастным категориям. Это поможет понять закономерность, между возрастным ограничением книги и количеством страниц в ней, что может быть полезно для исследований. После нажатия кнопки “перейти на следующий уровень иерархии” (рис. 8) выводится следующая круговая диаграмма - среднее значение страниц в книгах по Издательствам (рис. 9). Это поможет изучить информацию о том, к какому издательству лучше обращаться за длинными или короткими книгами. При нажатии на конкретный сегмент круговой диаграммы, можно подробнее изучить, среднее количество страниц в книгах, изданных соответствующим издательством. (рис. 10)

### 6.3 Информационная панель в Excel с графиком и "срезам"

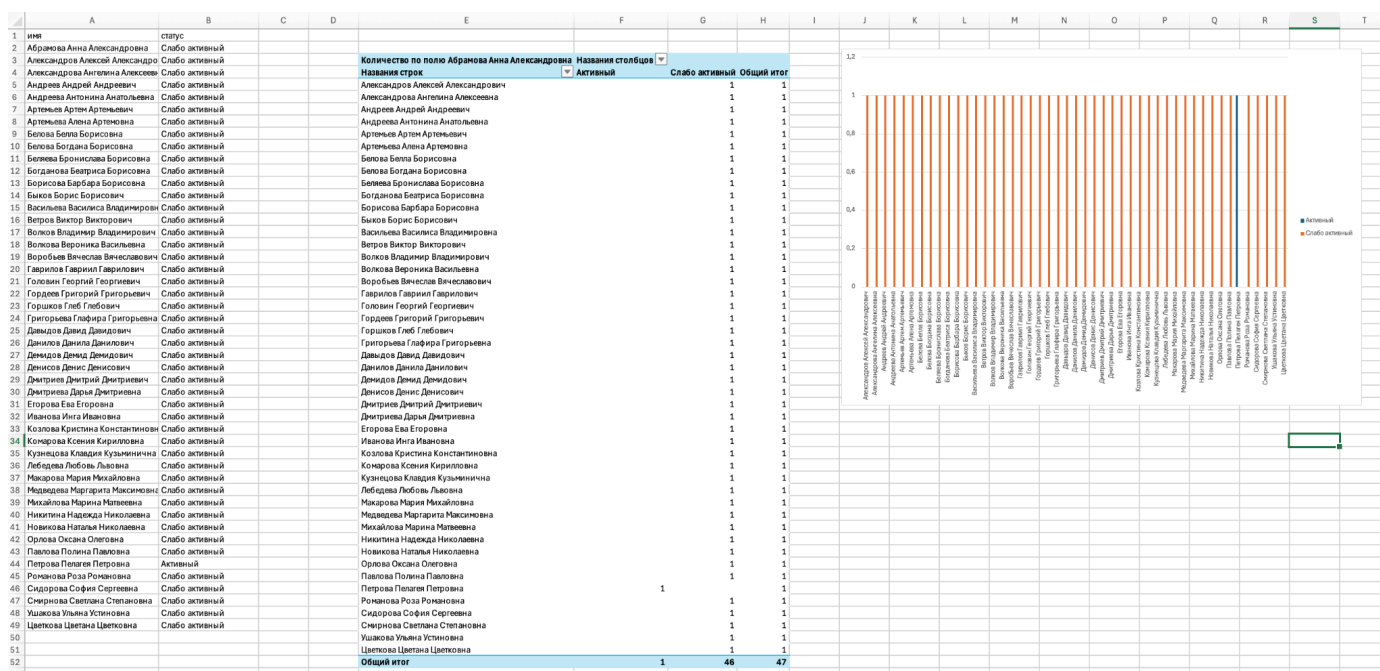


Рис. 11. Отчет в Excel.

## Результаты анализа

- **Слабо активные читатели:** Большинство читателей (49 из 50) попадают в категорию "Слабо активный", что указывает на низкий уровень вовлеченности. Эти читатели взяли не более двух книг за последние шесть месяцев.
- **Активные читатели:** Только один читатель был классифицирован как "Активный", взяв от трех до пяти книг за последние шесть месяцев. Это указывает на средний уровень вовлеченности.

Такие результаты указывают на слабый уровень вовлеченности читателей. Значит, нужно разработать новые стратегии привлечения клиентов.

## **7. Вывод**

В результате проделанной работы была разработана база данных, которая предназначена для управления книжными фондами библиотеки и ее читателями. Данная база данных обеспечивает не только детализированный учет всех книг в библиотеке, но и контроль за процессами выдачи и возврата, что значительно упрощает работу сотрудников библиотеки. Кроме того, есть возможность отслеживать визиты читателей в библиотеку, что важно для соблюдения мер безопасности.

База данных дает простор для проведения различных анализов над данными о книгах, читателях, авторах, жанрах и др. Есть возможность получать структурированные отчеты для осуществления управления библиотекой.

Мы считаем разработанную базу данных эффективной и отвечающей поставленным задачам. Все участники показали высокую степень владения необходимыми для разработки навыками и активно участвовали в выполнении задания.

## 8. Описание вклада участников в проект

Задачи	Дарья	Диана	Карина	Екатери на	Юлия
Анализ и описание предметной области	+	+	+	+	+
Разработка модели предметной области (инфологической модели) НОТАЦИЯ ЧЕНА	+	+	+	+	+
Разработка концептуальной модели данных (даталогической модели) IDEF1X	+	+	+	+	+
Размещение базы данных в СУБД	+	+	+	+	+
Заполнение базы данных тестовыми данными	+	+	+	+	+
Настройка индексов	+				
Написание SQL запросов	+	+	+	+	+
Процедуры	+	+			
Триггеры			+		
Функции	+				
Представления					+
Power BI отчеты	+		+	+	+
Написание отчета по проделанной работе	+	+	+	+	+
Видеоролик (съемки + участие)		+			

## 9. Приложения

Скрипт для создания базы данных в MS SQL Server.

```
CREATE TABLE Reader
(
    id_reader INT IDENTITY(1,1) ,
    reader_name NVARCHAR(100) NOT NULL,
    age INT NOT NULL,
    deposit_balance SMALLMONEY DEFAULT 0,
    PRIMARY KEY (id_reader)
);
CREATE TABLE Author
(
    id_author INT IDENTITY(1,1) ,
    author_name NVARCHAR(100) NOT NULL,
    PRIMARY KEY (id_author)
);
CREATE TABLE Genre
(
    id_genre INT IDENTITY(1,1) ,
    genre_name NVARCHAR(100) NOT NULL,
    PRIMARY KEY (id_genre)
);
CREATE TABLE Publishing_house
(
    id_publishing_house INT IDENTITY(1,1) ,
    publishing_house_name NVARCHAR(100) NOT NULL,
    PRIMARY KEY (id_publishing_house)
);
CREATE TABLE Visit
(
    id_visit INT IDENTITY(1,1) ,
    time_of_visit DATETIME DEFAULT GETDATE() ,
    id_reader INT NOT NULL,
    PRIMARY KEY (id_visit),
    FOREIGN KEY (id_reader) REFERENCES Reader(id_reader)
);
CREATE TABLE Language
(
    id_language INT IDENTITY(1,1) ,
    language_name NVARCHAR(100) NOT NULL,
    PRIMARY KEY (id_language)
);
```

```

CREATE TABLE Country
(
    id_country INT IDENTITY(1,1),
    country_name NVARCHAR(100) NOT NULL,
    PRIMARY KEY (id_country)
);

CREATE TABLE Contact_details
(
    id_contact_details INT IDENTITY(1,1),
    phone_number NVARCHAR(50) NOT NULL,
    email NVARCHAR(100) NOT NULL,
    address NVARCHAR(200) NOT NULL,
    id_reader INT NOT NULL,
    PRIMARY KEY (id_contact_details),
    FOREIGN KEY (id_reader) REFERENCES Reader(id_reader)
);

CREATE TABLE Age_category
(
    id_category VARCHAR(3) NOT NULL,
    PRIMARY KEY (id_category),
);

CREATE TABLE Book
(
    id_book INT IDENTITY(1, 1),
    name_book NVARCHAR(150) NOT NULL,
    year_of_publishing INT NOT NULL,
    id_category VARCHAR(3) NOT NULL,
    status BIT DEFAULT 1,
    number_of_pages INT NOT NULL,
    id_language INT NOT NULL,
    id_country INT NOT NULL,
    id_publishing_house INT NOT NULL,
    PRIMARY KEY (id_book),
    FOREIGN KEY (id_language) REFERENCES Language(id_language),
    FOREIGN KEY (id_country) REFERENCES Country(id_country),
    FOREIGN KEY (id_publishing_house) REFERENCES
Publishing_house(id_publishing_house),
    FOREIGN KEY (id_category) REFERENCES
Age_category(id_category)
);

CREATE TABLE Loan
(
    id_loan INT IDENTITY(1,1),
    start_date DATE DEFAULT CAST(GETDATE() AS DATE),

```

```

tentative_return_date DATE NOT NULL,
real_return_date DATE,
id_reader INT NOT NULL,
id_book INT NOT NULL,
PRIMARY KEY (id_loan),
FOREIGN KEY (id_reader) REFERENCES Reader(id_reader),
FOREIGN KEY (id_book) REFERENCES Book(id_book)
);
CREATE TABLE Review
(
    id_review INT IDENTITY(1,1),
    review_date DATETIME DEFAULT GETDATE(),
    rating INT NOT NULL,
    review_text NTEXT,
    id_reader INT NOT NULL,
    id_book INT NOT NULL,
    PRIMARY KEY (id_review),
    FOREIGN KEY (id_reader) REFERENCES Reader(id_reader),
    FOREIGN KEY (id_book) REFERENCES Book(id_book)
);
CREATE TABLE book_has_author
(
    id_author INT NOT NULL,
    id_book INT NOT NULL,
    PRIMARY KEY (id_author, id_book),
    FOREIGN KEY (id_author) REFERENCES Author(id_author),
    FOREIGN KEY (id_book) REFERENCES Book(id_book)
);
CREATE TABLE book_has_genre
(
    id_genre INT NOT NULL,
    id_book INT NOT NULL,
    PRIMARY KEY (id_genre, id_book),
    FOREIGN KEY (id_genre) REFERENCES Genre(id_genre),
    FOREIGN KEY (id_book) REFERENCES Book(id_book)
);

```

