Величко Т.Д ИТТС-о-22/1

Задание по практике

Дано:

Мобильное клиент-серверное приложение «Список покупок в магазинах» (Примечание: приложение без интеграции с магазинами).

Необходимо описать:

1. Бизнес-логику мобильного приложения (использовать любую удобную нотацию моделирования бизнес-процессов).

2. Основные функции данного приложения (список должен быть составлен в порядке убывания важности функций для пользователя).

3. Процесс синхронизации данных между клиентом и сервером (добавление и удаление списка, наполнение и редактирование списка, покупка/«откупка продукта» и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

4. Подготовить прототип одного из экранов мобильного приложения и описать пользовательский интерфейс для данного экрана (например, покупка товара).

5. Подготовить подробное описание функции покупки товара, которую можно было бы использовать в качестве постановки задачи для разработки (помимо текстового описания использовать UML диаграммы, указать используемые API методы, передаваемые и получаемые параметры, описать процесс хранения информации о покупках пользователя).

6. Основные, на ваш взгляд, сложности разработки такого приложения. Вопросы, возникшие при выполнении тестового задания, которые вы бы задали заказчику.

7. Есть таблицы Books и Authors, где AuthorId табл. Books равно Id табл. Authors.



Необходимо написать SQL-запрос, чтобы найти:

1. Общую стоимость книг для каждого автора и отсортировать результат в порядке убывания;

2. Стоимость книг автора превышает 1500;

3. Вывести авторов с количеством книг;

4. Получить автора без книг.

**Сдача задания:**

Задания необходимо выполнить в электронном текстовом документе.

К диаграммам дополнительно прикрепить ссылку на файл в формате .jpeg, .png. При отправке ссылок на файлы необходимо предусмотреть, чтобы документы были доступны для просмотра проверяющими.

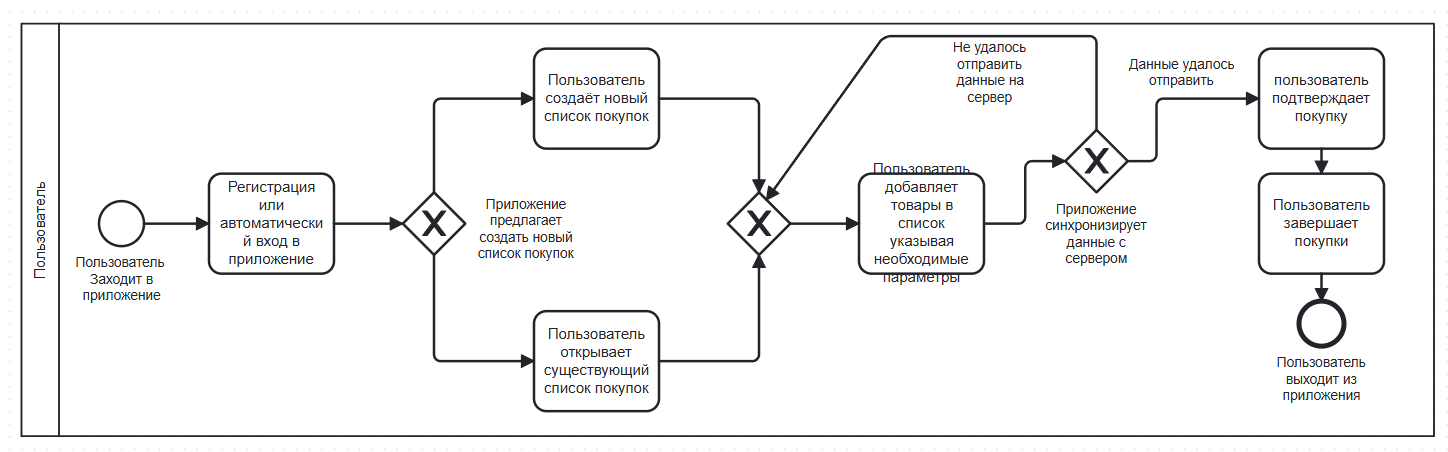
Для макетов также необходима ссылка на проект + ссылки на изображения формата .jpeg, .png

**Выполнение тестового задания**

1. Бизнес-логику мобильного приложения (использовать любую удобную нотацию моделирования бизнес-процессов).

Ответ:

Пример бизнес-логики для приложения:

Пользователь открывает приложение и создает новый список покупок.  
Пользователь может добавлять товары в список, редактировать их или удалять.  
Приложение синхронизирует список покупок с сервером, чтобы данные были доступны на разных устройствах.  
Пользователь может отмечать товары для подтверждения покупок.  
Приложение отслеживает историю покупок, количество оставшихся товаров и уведомляет пользователя о необходимых покупках.  


Ссылка в лучшем разрешении

Svg:

[diagram.svg](https://drive.google.com/file/d/1UsulYw3e0y1eXow8lEEXdEx-GdOWvguY/view?usp=sharing)

Png:

<https://drive.google.com/file/d/11PyyhNPG5n3W-RibWc9L1dFsVOjGVt7j/view?usp=sharing>

2. Основные функции данного приложения (список должен быть составлен в порядке убывания важности функций для пользователя).

Ответ:

Список основных функций приложения по убыванию важности:

**Создание и редактирование списка покупок** Пользователь может создавать новые списки покупок, редактировать их, добавлять или удалять товары, что является основной функцией приложения.  
**Отметка товаров как купленных**

После того как товары были приобретены, пользователь может пометить их как купленные, чтобы избежать их повторного добавления в список.  
**Синхронизация данных между клиентом и сервером** Важно, чтобы данные о списках покупок синхронизировались между устройствами  
**Просмотр истории покупок** Пользователь может просматривать список всех сделанных покупок, что может быть полезно для анализа затрат или для восстановления ранее купленных товаров.  
**Напоминания о необходимых покупках** Приложение может напоминать пользователю о товарах, которые нужно купить, основываясь на их предпочтениях или на истории покупок.

**Поиск и фильтрация товаров** Возможность искать товары по наименованию, категории, цене или другим критериям.  
**Подсчет общей стоимости товаров в списке** Приложение может автоматически подсчитывать стоимость всех товаров в списке, что помогает пользователю контролировать бюджет.

**Уведомления о скидках и акциях** Приложение может информировать пользователя о скидках на товары из его списка

3. Процесс синхронизации данных между клиентом и сервером (добавление и удаление списка, наполнение и редактирование списка, покупка/«откупка продукта» и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

Ответ:

### **Процесс синхронизации данных между клиентом и сервером Процесс синхронизации данных**

**1) Добавление товара в список**:

**Действия пользователя:** Пользователь добавляет товар в список покупок на мобильном устройстве.

**Действия клиента:** Клиент отправляет запрос на сервер с данными о товаре .

**Синхронизация сервера:** Сервер сохраняет товар в базе данных и возвращает подтверждение успешного добавления.

**2) Удаление товара из списка**:

**Действия пользователя:** Пользователь удаляет товар из списка покупок.

**Действия клиента:** Клиент отправляет запрос на сервер с идентификатором удаляемого товара.

**Синхронизация сервера:** Сервер удаляет товар из базы данных и возвращает успешный ответ.

#### **3) Редактирование списка товаров**

**Действия пользователя:** Пользователь может изменить количество или другие характеристики товара. Для этого делает соответствующие действия в приложения.

**Действия клиента:** Клиент отправляет обновленные данные на сервер.

**Синхронизация сервера:** Сервер обновляет информацию в базе данных.и отправляет обратно ответ.

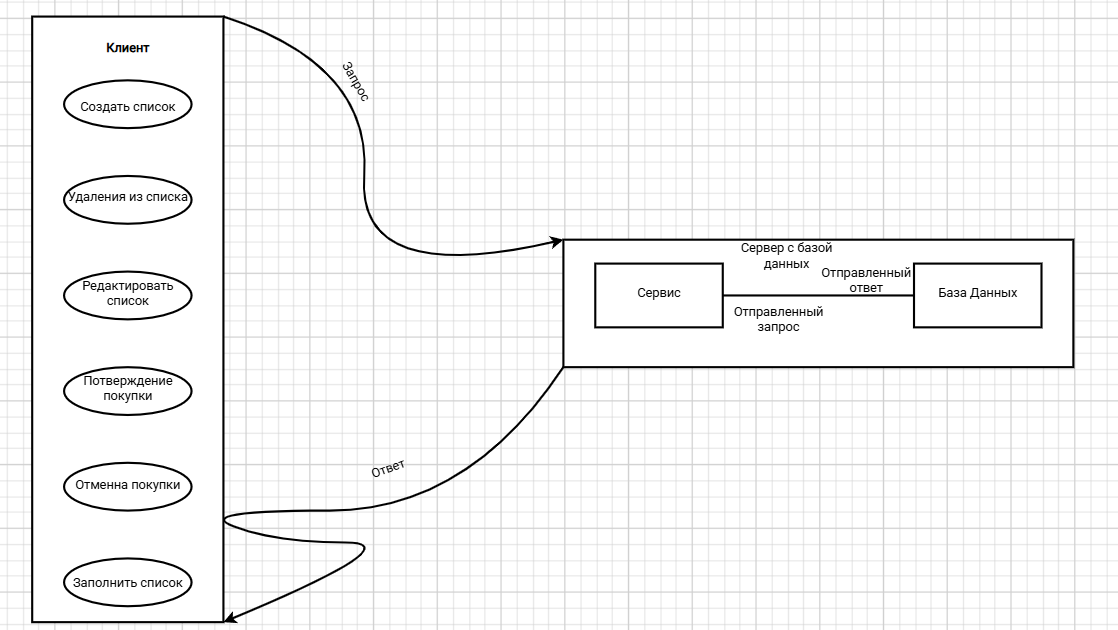
#### **4) Покупка товара (пометка товара как купленный)**

**Действия пользователя:** Пользователь отмечает товар как купленный.

**Действия клиента:** Клиент отправляет запрос на сервер, что товар был куплен, возможно, с уточнением времени покупки и других деталей.

**Синхронизация сервера:** Сервер обновляет статус товара в базе данных.Затем отправляет ответ в клиент.

**Диаграмма UML**



Ссылка в гугл диск: <https://drive.google.com/file/d/1HLQ0nSwGnYzSwGfR76aUPVojTsuvznQ-/view?usp=sharing>

**API Метод**

**POST /lists/{list\_id}/items –** добавление нового товара в список

Параметры запроса:  
{  
 "product\_name": "detergent",  
 "quantity": 2,  
 "category": "Chemistry"  
 }

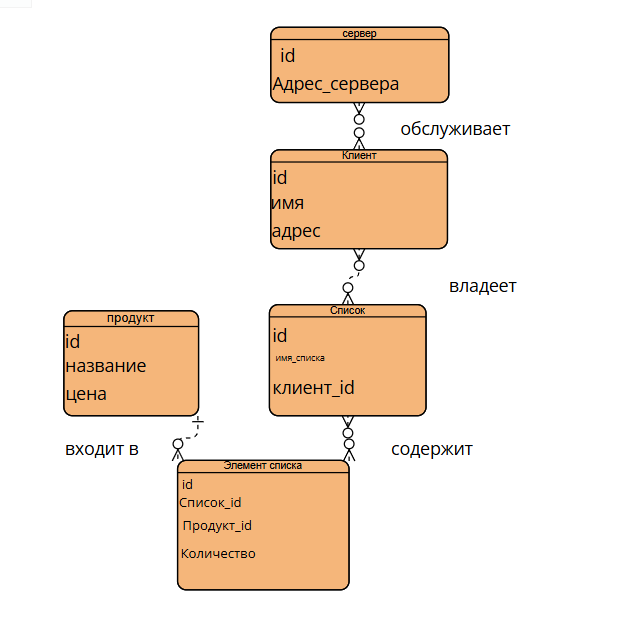
Ответ:

{  
 "status": "success",  
 "message": "Item added successfully",  
 "item": {  
 "product\_id": 543,  
 "product\_name": "detergent",  
 "quantity": 2,  
 "category": "Chemistry"  
 }  
 }  
  
**DELETE /lists/{list\_id}/items/{item\_id}** – удаление товара из списка

Параметры запроса:  
{  
 "item\_id": 543  
 }  
Ответ:   
{  
 "status": "success",  
 "message": "Item removed successfully"  
 }  
  
**PUT /lists/{list\_id}/items/{item\_id}** – обновление количества товара

Параметры запроса:  
{  
 "quantity": 3  
 }  
Ответ:   
{  
 "status": "success",  
 "message": "Item updated successfully",  
 "item": {  
 "product\_id": 543,  
 "quantity": 3  
 }  
 }  
  
**POST /lists/{list\_id}/items/{item\_id}/purchase** – покупка товара (пометка как купленный)

Параметры запроса:  
{  
 "status": "purchased"  
 }  
Ответ:   
{  
 "status": "success",  
 "message": "Item marked as purchased"  
 }  
**ER-диаграмма**



ссылка в гугл:https://drive.google.com/file/d/1w2EQthOuNYDUpFsHFUClIl4ZdKu2QOKe/view?usp=sharing

4. Подготовить прототип одного из экранов мобильного приложения и описать пользовательский интерфейс для данного экрана

(покупка товара)

Экран прототип экрана **"Покупка товара"** позволяет пользователю увидеть список товаров, которые он добавил в свой список покупок, а также отмечать товары как купленные. Основное действие на экране — это пометка товара как купленного.

**Основные элементы интерфейса:**

**Общее количество товаров** — отображается общее количество товаров в списке и количество купленных.

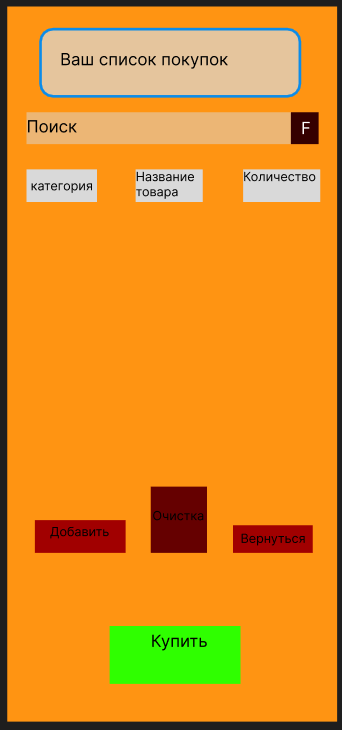
**Фильтры по категориям** — возможность фильтровать товары по категориям (например, "Бытовая химия", "Одежда").

**Список товаров** — на экране отображается список всех товаров, добавленных пользователем в список покупок.

**Товар (строка)** — каждая строка в списке представляет товар с его названием, количеством, категорией и кнопкой для пометки как купленного.

**Кнопка "Купить"** — кнопка на каждой строке, которая позволяет пометить товар как купленный. При нажатии товар становится серым или получает отметку "Куплено".

**Поиск по товарам** — строка для поиска товаров по названию.



Ссылка на фигму макет:: https://www.figma.com/design/tsFy0bD9i39M1nvuCq60ob/praktika?node-id=0-1&t=Ii8mtqy0DinKx1sO-1

### **Пояснение элементов:**

**Кнопки навигации:  
Вернуться** — для выхода на предыдущий экран.

**Очистить покупки** — удаление всех купленных товаров из списка.

**Добавить товар** — добавление нового товара в список покупок.

**Список товаров:** В каждой строке отображаются: Название товара.Количество товара в списке. Категория товара (например, молочные продукты).

**Кнопка для пометки товара как купленного.** Эта кнопка меняет свой цвет и текст на "Куплено", если товар уже был куплен.

**Поиск и фильтры:**Строка поиска позволяет найти товар по имени.  
Фильтры помогают отфильтровать товары по категориям, например, только химия или бытовая химия..

**Общее количество и купленные товары:**

Внизу экрана отображается количество всех товаров в списке и количество тех, которые уже были куплены. Это позволяет пользователю следить за прогрессом.

**5. Подготовить подробное описание функции покупки товара, которую можно было бы использовать в качестве постановки задачи для разработки**

### **Общее описание:**

Функция покупки товара позволяет пользователю пометить товар в своем списке покупок как купленный. После того как товар помечен как купленный, его статус обновляется на сервере и синхронизируется с клиентом. Это позволяет пользователю отслеживать, какие товары были куплены, а какие еще нужно приобрести.

### **Шаги выполнения функции:**

**Инициация покупки товара (пометка как купленный):**

Пользователь нажимает кнопку "Купить" рядом с товаром в списке покупок.

Приложение отправляет запрос на сервер с информацией о товаре, который был выбран для пометки как купленный.

Сервер обновляет статус товара в базе данных и помечает его как "купленный".

**Обновление состояния на клиенте:**

После получения подтверждения от сервера, клиент обновляет отображение товара:

Статус товара меняется на "Куплено".

Кнопка "Купить" становится неактивной или заменяется на надпись "Куплено".

Система пересчитывает количество купленных товаров и отображает обновленную информацию пользователю.

**Синхронизация с сервером:**

Сервер сохраняет данные о купленных товарах и синхронизирует их с другими устройствами пользователя, если он использует несколько устройств для работы с приложением.

### **API методы:**

**POST /lists/{list\_id}/items/{item\_id}/purchase** – Покупка товара (пометка как купленный)

**Описание:** Этот метод используется для того, чтобы пометить товар как купленный.  
**Параметры запроса:**

{

"status": "purchased"

}

**Ответ:**

{

"status": "success",

"message": "Item marked as purchased",

"item": {

"product\_id": 543,

"product\_name": "detergent",

"quantity": 2,

"category": "chemistry",

"status": "purchased"

}

}

**GET /lists/{list\_id}/items** – Получение списка товаров

**Описание:** Этот метод используется для получения всех товаров из списка покупок, включая их статус (куплены или нет).  
**Ответ:**{

"status": "success",

"items": [

{

"product\_id": 543,

"product\_name": "detergent",

"quantity": 2,

"category": "chemistry",

"status": "purchased"

},

{

"product\_id": 242,

"product\_name": "sponge",

"quantity": 3,

"category": "household goods",

"status": "not\_purchased"

}

]

}

### **Параметры и хранилище данных:**

**Таблица Products (База данных)** Для хранения информации о товарах, их статусе и количестве используется таблица Products. При изменении статуса товара на "купленный", это поле обновляется в базе данных.  
  
 **Пример таблицы Products:**

| product\_id | product\_name | quantity | category | status |
| --- | --- | --- | --- | --- |
| 543 | detergent | 2 | chemistry | purchased |
| 242 | sponge | 3 | household goods | not\_purchased |

**product\_id:** Уникальный идентификатор товара (для каждого товара).

**product\_name:** Название товара.

**quantity:** Количество товара в списке.

**category:** Категория товара (например, "Бытовая химия или товары для дома").

**status:** Статус товара (например, "purchased" или "not\_purchased").

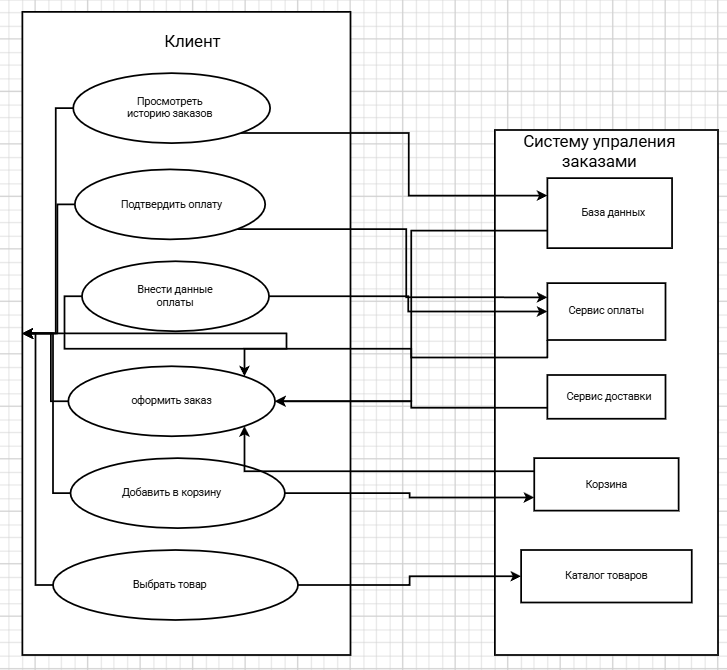
**Обновление состояния на сервере:** Когда товар помечен как купленный, информация о нем обновляется на сервере, и синхронизация с клиентами происходит в реальном времени.

### **Процесс хранения информации о покупках пользователя:**

Вся информация о покупках хранится в базе данных, и пользователи могут отслеживать, какие товары они уже купили, а какие еще нужно купить.

Когда пользователь помечает товар как купленный, его статус в базе данных изменяется на "purchased". Это изменение сохраняется на сервере и передается обратно на клиент.

Любые действия синхронизируются с сервером, что позволяет пользователю работать с несколькими устройствами (например, мобильный телефон и планшет).



6 Основные, на ваш взгляд, сложности разработки такого приложения. Вопросы, возникшие при выполнении тестового задания, которые вы бы задали заказчику.

**Синхронизация данных между клиентом и сервером**:

Одной из главных сложностей является обеспечение правильной и своевременной синхронизации данных между мобильным клиентом и сервером. Это важно для обеспечения актуальности данных о списке покупок, а также для правильного отображения статусов товаров (например, покупка товара).

Как часто должна происходить синхронизация данных? Как обрабатывать ситуации, когда клиент теряет связь с сервером (например, при нестабильном интернете)?

**Обработка состояния товаров**:  
Потребуется продумать, как хранить и обновлять информацию о товарах, статусах (куплены/не куплены) и количестве. Важно, чтобы информация обновлялась корректно и на клиенте, и на сервере, а также была синхронизирована между устройствами.

**Вопросы заказчику**: Как будут обрабатываться случаи, когда пользователь отменяет покупку товара (возврат)? Нужно ли предусматривать возможность частичной покупки товаров (например, если покупка товарного набора не завершена полностью)?

**Обработка ошибок при взаимодействии с сервером**:  
Важно правильно обрабатывать возможные ошибки при взаимодействии с сервером, такие как неудачная синхронизация, отсутствие интернет-соединения, ошибки в API и другие.

**Вопросы заказчику**: Какие ошибки следует предусмотреть в приложении? Как часто нужно показывать пользователю уведомления о проблемах с синхронизацией или сервером?

**Производительность и масштабируемость**:  
Приложение должно работать быстро и корректно на разных устройствах. С увеличением количества товаров в списке и пользователей, потребуется внимание к производительности сервера и мобильного приложения.

**Вопросы заказчику**: Какие ожидания по производительности приложения? Сколько товаров будет в среднем в списке покупок, и сколько пользователей ожидается?

**Проблемы с безопасностью**:  
Обработка личных данных пользователя и синхронизация их с сервером требует обеспечения безопасности. Важно защитить данные о пользователях, их покупках и другой личной информации.

**Вопросы заказчику**: Какую систему аутентификации вы хотите использовать для приложения? Нужно ли реализовывать шифрование данных между клиентом и сервером?

**Интерфейс и опыт пользователя**:  
Очень важно, чтобы интерфейс был удобным и интуитивно понятным, особенно когда речь идет о таких функциональных элементах, как покупка товара и синхронизация данных.

**Вопросы заказчику**: Есть ли у вас предпочтения по дизайну интерфейса (например, стиль, цвета)? Нужно ли предусматривать дополнительные функции, такие как уведомления о скидках или акциях на товары?

**Поддержка различных устройств и платформ**:  
Приложение должно быть совместимо с различными устройствами и платформами (Android, iOS), что может вызывать дополнительные сложности в процессе разработки.

**Вопросы заказчику**: Для каких платформ должно быть разработано приложение (iOS, Android)? Есть ли планы на расширение функционала для других платформ в будущем?

7. Есть таблицы Books и Authors, где AuthorId табл. Books равно Id табл. Authors.

Необходимо написать SQL-запрос, чтобы найти: 1. Общую стоимость книг для каждого автора и отсортировать результат в порядке убывания; 2. Стоимость книг автора превышает 1500; 3. Вывести авторов с количеством книг; 4. Получить автора без книг.

#### **1. Общая стоимость книг для каждого автора и сортировка по убыванию:**

**Запрос:**

SELECT

a.Name AS AuthorName,

SUM(b.Price) AS TotalBookPrice

FROM

Books b

JOIN

Authors a ON b.AuthorId = a.Id

GROUP BY

a.Name

ORDER BY

TotalBookPrice DESC;

#### **2. Поиск авторов, чья стоимость книг превышает 1500:**

**Запрос:**

SELECT

a.Name AS AuthorName,

SUM(b.Price) AS TotalBookPrice

FROM

Books b

JOIN

Authors a ON b.AuthorId = a.Id

GROUP BY

a.Name

HAVING

SUM(b.Price) > 1500;

#### **3. Вывод авторов с количеством книг:**

**Запрос:**

SELECT

a.Name AS AuthorName,

COUNT(b.Id) AS NumberOfBooks

FROM

Books b

JOIN

Authors a ON b.AuthorId = a.Id

GROUP BY

a.Name;

#### **4. Вывод авторов без книг:**

**Запрос:**

SELECT

a.Name AS AuthorName

FROM

Authors a

LEFT JOIN

Books b ON a.Id = b.AuthorId

WHERE

b.Id IS NULL;