Нужно выбрать любой интернет-магазин, где есть процесс оформления заказа *(Ozon, Delivery club и т.д.)*
Все задания будем делать по выбранному сервису.

Задание 1 (Бизнес-процессы)

Разработать схему процесса оформления заказа.
Нотация – любая.

Процесс оформления заказа на сайте dns-shop.ru
Activity diagram

Задание 2 (База данных)

Смоделировать структуру базы данных ERD на уровне физической модели.

В диаграмме необходимо отразить все сущности, которые участвуют в процессе оформления заказа.

По каждой таблице нужно указать краткое текстовое описание, какие данные хранятся в данной таблице.
Например: таблица User - в ней храним данные о пользователях, и т.п.

Также необходимо дать описание для **каждого** поля в таблице.
Например:

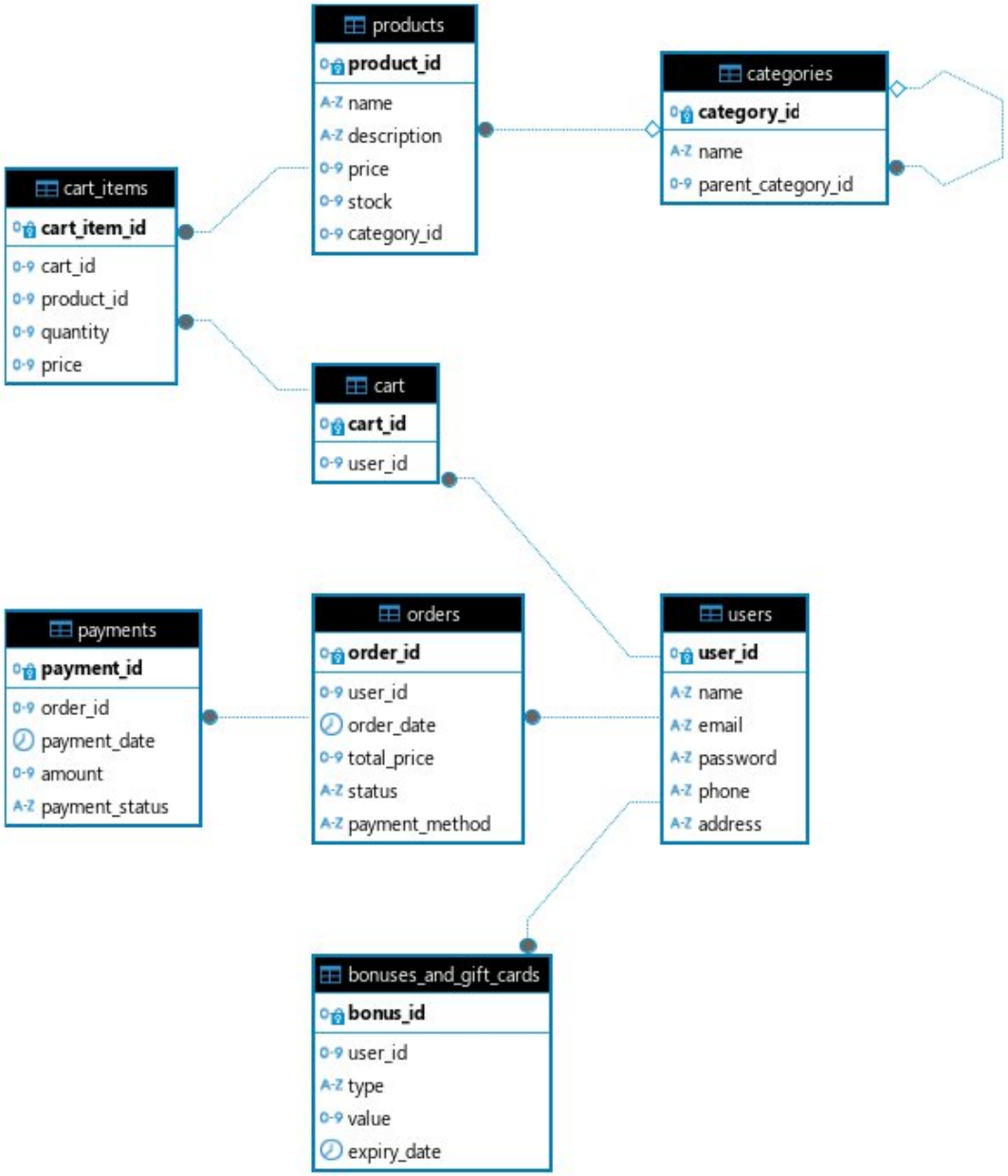
Поле	**Описание**
payment_date	Дата оплаты заказа

* Напишите запрос, с помощью которого можно получить данные о пользователях и их заказах за указанный период *(например, за последние полгода)*.
Учитываем только те, которые пользователь выкупил.
Результат запрос должен вернуть информацию в следующем виде:

Фамилия Имя пользователя	**ID пользователя**	**Кол-во заказов**	**Сумма заказов**

К запросу **обязательно** приложить:
1. Дамп базы данных, которая изображена на ER-диаграмме. Он должен быть выполнен в СУБД PostgreSQL.
2. Скрин успешно выполненного запроса.

ER-диаграмма



Таблицы:

users — хранит данные о пользователях.
categories — категории товаров, подкатегории.
products — информация о товарах.
cart — корзина покупок пользователей.
cart_items — детали товаров в корзине.
orders — информация о заказах.

Описание полей:

users

user_id: уникальный идентификатор пользователя.
name: имя пользователя.
email: электронная почта

password:пароль
phone: номер телефона.
address: адрес пользователя.

categories

category_id: уникальный идентификатор категории.
name: название категории.

products

product_id: уникальный идентификатор товара.
name: название товара.
description: описание товара.
price: цена товара.
stock: количество товаров на складе.
category_id:ссылка на категорию товара.

cart

cart_id: уникальный идентификатор корзины.
user_id: ссылка на пользователя.

cart_items

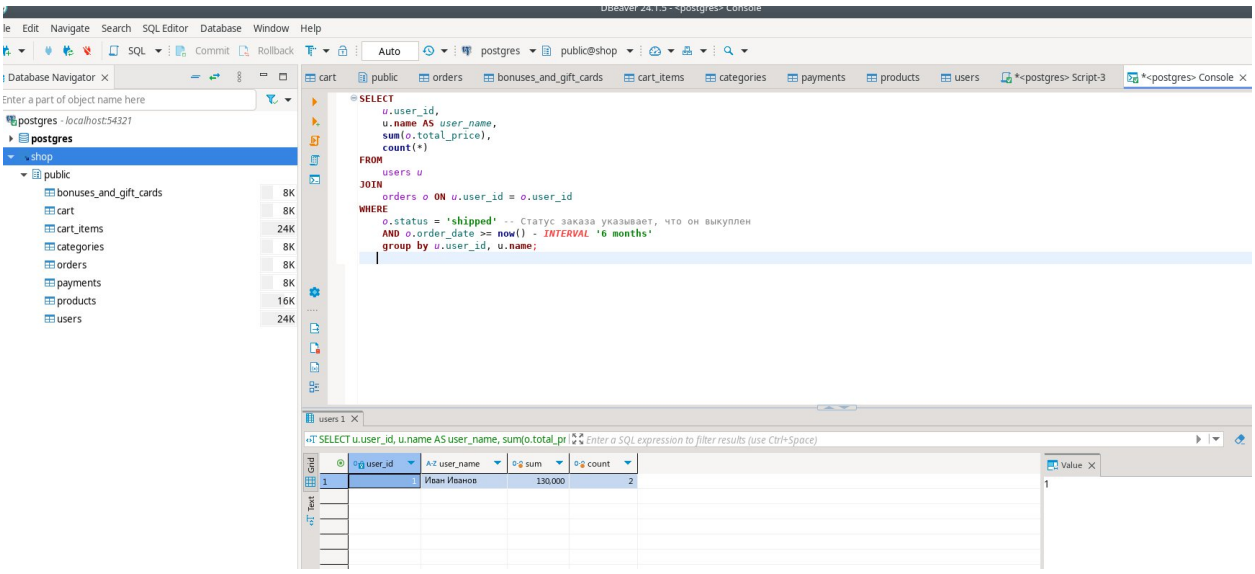
cart_item_id: уникальный идентификатор элемента.
cart_id: ссылка на корзину.
product_id: ссылка на товар.
quantity: количество товара.
price: цена товара

orders

order_id: уникальный идентификатор заказа.
user_id: ссылка на пользователя.
order_date: дата заказа.
status: статус заказа
payment_method: способ оплаты
total_price: итоговая цена заказа.

Запрос

```sql
SELECT
    u.user_id,
    u.name AS user_name,
    sum(o.total_price),
    count(*)
FROM
    users u
JOIN
    orders o ON u.user_id = o.user_id
WHERE
    o.status = 'shipped' -- Статус заказа указывает, что он выкуплен
    AND o.order_date >= now() - INTERVAL '6 months'
    group by u.user_id, u.name;
```


DBeaver 24.1.5 - <postgres> Console
Edit Navigate Search SQL Editor Database Window Help
SQL Commit Rollback Auto postgres public@shop
Database Navigator
Enter a part of object name here
postgres - localhost:54321
postgres
shop
public
bonuses_and_gift_cards 8K
cart 8K
cart_items 24K
categories 8K
orders 8K
payments 8K
products 16K
users 24K
cart public orders bonuses_and_gift_cards cart_items categories payments products users *<postgres> Script-3 *<postgres> Console
SELECT
u.user_id,
u.name AS user_name,
sum(o.total_price),
count(*)
FROM
users u
JOIN
orders o ON u.user_id = o.user_id
WHERE
o.status = 'shipped' -- Статус заказа указывает, что он выкуплен
AND o.order_date >= now() - INTERVAL '6 months'
group by u.user_id, u.name;
users 1
SELECT u.user_id, u.name AS user_name, sum(o.total_pr
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
user_id user_name sum count
1 1 Иван Иванов 130,000 2
Value
1

Задание 3 (API)

Описать API для процессов:

- Создание заказа
- Получение детальной информации по конкретному заказу
- Получение информации по всем заказам для конкретного пользователя

В описании каждого метода API должно быть:
1) HTTP-метод,
2) URL-адрес
3) описание входных параметров *(тип, обязательность)* в формате JSON
4) описание выходных параметров *(тип, обязательность)* в формате JSON
5) маппинг с полями БД *(для **каждого** атрибута в запросе обязательно указать, какому полю в БД он соответсвует)*.

* Описать API в Swagger *(результат прислать в файле)*.

1. Создание заказа

HTTP-метод

POST

URL-адрес

/api/orders

Описание входных параметров

```
{
 "user_id": {
   "type": "integer",
   "required": true
 },
 "cart_id": {
   "type": "integer",
   "required": true
 },
 "payment_method": {
   "type": "string",
   "required": true
 }
}
```

Описание выходных параметров

```
{
 "order_id": {
   "type": "integer",
```

```
    "required": true
  },
  "order_date": {
    "type": "string",
    "format": "date-time",
    "required": true
  },
  "status": {
    "type": "string",
    "required": true
  },
  "total_price": {
    "type": "number",
    "required": true
  }
}
```

Маппинг с полями БД

• user_id → user_id (таблица orders)

• cart_id → cart_id (для расчета итоговой цены и создания заказа)

• payment_method → payment_method (таблица orders)

• order_date → автоматически устанавливается на текущую дату (таблица orders)

• status → устанавливается в значение по умолчанию (например, "pending") (таблица orders)

• total_price → вычисляется на основе элементов корзины и сохраняется в поле total_price (таблица orders)

2. Получение детальной информации по конкретному заказу

HTTP-метод

GET

URL-адрес

/api/orders/{order_id}

Описание входных параметров

```
{
  "order_id": {
    "type": "integer",
    "required": true
  }
```

}

Описание выходных параметров


```
{
 "order_id": {
  "type": "integer",
  "required": true
 },
 "user_id": {
  "type": "integer",
  "required": true
 },
 "order_date": {
  "type": "string",
  "format": "date-time",
  "required": true
 },
 "status": {
  "type": "string",
  "required": true
 },
 "payment_method": {
  "type": "string",
  "required": true
 },
 "total_price": {
  "type": "number",
  "required": true
 },
 "items": {
  "type": "array",
  "required": true,
  "items": {
   "product_id": {
    "type": "integer",
    "required": true
   },
   "quantity": {
    "type": "integer",
    "required": true
   },
   "price": {
    "type": "number",
    "required": true
   }
  }
 }
}
```


Маппинг с полями БД

• order_id → order_id (таблица orders)

• user_id → user_id (таблица orders)

• order_date → order_date (таблица orders)

• status → status (таблица orders)

• payment_method → payment_method (таблица orders)

• total_price → total_price (таблица orders)

• items → данные извлекаются из таблицы cart_items, связывая их по полю cart_id, которое соответствует заказу.

3. Получение информации по всем заказам для конкретного пользователя

HTTP-метод

GET

URL-адрес

/api/users/{user_id}/orders

Описание входных параметров

```
{
  "user_id": {
    "type": "integer",
    "required": true
  }
}
```

Описание выходных параметров

```
{
  "orders": {
    "type": "array",
    "required": true,
    "items": {
      "order_id": {
        "type": "integer",
        "required": true
      },
      "order_date": {
        "type": "string",
        "format": "date-time",
        "required": true
      },
```

```
      "status": {
        "type": "string",
        "required": true
      },
      "total_price": {
        "type": "number",
        "required": true
      }
    }
  }
}
```

Маппинг с полями БД

• user_id → user_id (фильтр для выборки из таблицы orders)

Задание 4 (Sequence diagram)

Предположим, в сервисе, который вы выбрали есть онлайн-оплата *(или эта функциональность действительно есть)*.

Нужно с помощью диаграммы последовательности отразить для процесса оплаты взаимодействие между нашим сервисом и сервисом оплаты.