

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет

«МИФИ»

Институт лазерных и плазменных технологий

Кафедра прикладной математики (№ 31)

Отчёт о работе по курсу

«Базы данных (теоретические основы баз данных)»

Вариант «Интернет магазин iHerb»

Выполнили	Светова В.С.
Группа	Б20-215
Вариант	Интернет магазин iHerb
Преподаватель	Павленко Д.А.

Москва 2022

Оглавление

1. Формулировка задания.....	3
2. Концептуальная модель базы данных.....	3
2.1. Конкретизация предметной области	3
2.2. Описание атрибутов.....	4
3. Логическое проектирование	5
4. Физическое проектирование	6
4.1. Создание таблиц.....	7
4.2. Заполнение базы данных	8
4.2.1. Подготовка данных	8
4.2.2. Программа заполнения базы данных.....	9
4.3. Результаты заполнения	13
5. Выполнение запросов.....	16

1. Формулировка задания.

Спроектировать базу данных для международного интернет-магазина iHerb, доставляющего в разные страны косметическую и лекарственную продукцию, представленную на их сайте. База данных должна содержать информацию о клиентах, пользующихся данным сервисом, о товарах, продаваемых на площадке и отзывы на них. Также база данных должна хранить информацию для доставки этих товаров.

2. Концептуальная модель базы данных.

В ходе выполнения задания был проведён анализ предметной области, после чего была спроектирована следующая концептуальная модель:

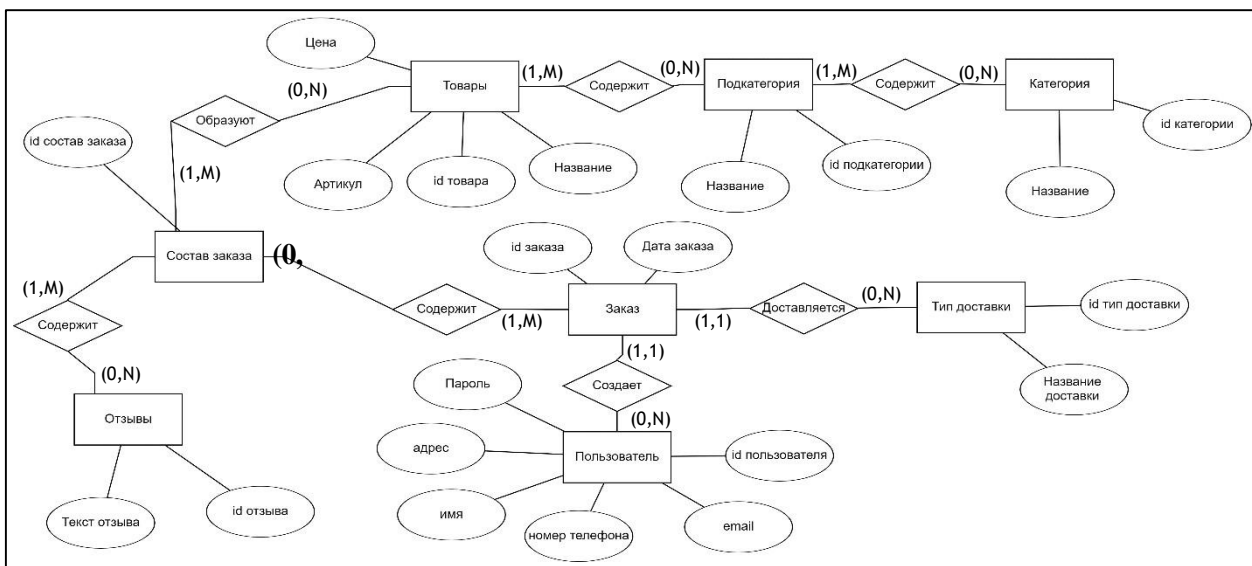


Рисунок 1 — концептуальная модель базы данных для предметной области интернет магазина iHerb.

2.1. Конкретизация предметной области

Необходимо базу данных, которая отражает основную информацию о товарах, продаваемых в интернет магазине iHerb. База данных должна также хранить информацию, необходимую для связи пользователя с покупаемыми им товарами. Позволять пользователям выбирать тип доставки. Также должны храниться отзывы на товары из сделанных заказов. Были выделены следующие сущности:

1. Категории(categories) – хранит информацию о категориях товаров представленных в интернет магазине.
2. Подкатегории(subcategories) – хранит информацию о подкатегориях товаров представленных в интернет магазине.
3. Товар (products) - хранит информацию о товарах представленных в интернет магазине.
4. Состав заказа (order_list) – позволяет связать заказ пользователя и товары находящиеся в данном заказе.
5. Заказы (orders) – хранит информацию о заказах, сделанных пользователями.
6. Отзывы (reviews) – хранит информацию о отзывах на товары в заказе.
7. Тип доставки (types_delivery) – хранит информацию о типах доставки.
8. Пользователь (users) – хранит информацию о пользователях.

2.2. Описание атрибутов

Для сущности категории (categories):

Атрибут	Расшифровка
categorie_id	Уникальный идентификатор категории
categorie_name	Название категории

Для сущности подкатегория (subcategories):

Атрибут	Расшифровка
subcategorie_id	Уникальный идентификатор подкатегории
subcategorie_name	Название подкатегории

Для сущности товар (products):

Атрибут	Расшифровка
product_id	Уникальный идентификатор товара
product_name	Название подкатегории
code	Артикул товара
price	Цена товара

Для сущности состав заказа (order_list):

Атрибут	Расшифровка
orderlist_id	Уникальный идентификатор состава заказа

Для сущности отзывы (reviews):

Атрибут	Расшифровка
reviews_id	Уникальный идентификатор отзыва
Created	Текст отзыва

Для сущности заказ (orders):

Атрибут	Расшифровка
order_id	Уникальный идентификатор заказа

data_order	Дата заказа
------------	-------------

Для сущности тип доставки (types_delivery):

Атрибут	Расшифровка
typedelivery_id	Уникальный идентификатор типа доставки
type_delivery	Тип доставки
delivery_cost	Цена доставки

Для сущности пользователь (users):

Атрибут	Расшифровка
user_id	Уникальный идентификатор пользователя
name	Имя пользователя
contact_phone	Номер телефона
password	Пароль
email	Электронная почта
adress	Адрес доставки

3. Логическое проектирование.

Далее, была разработана логическая модель базы данных, представленная ниже:

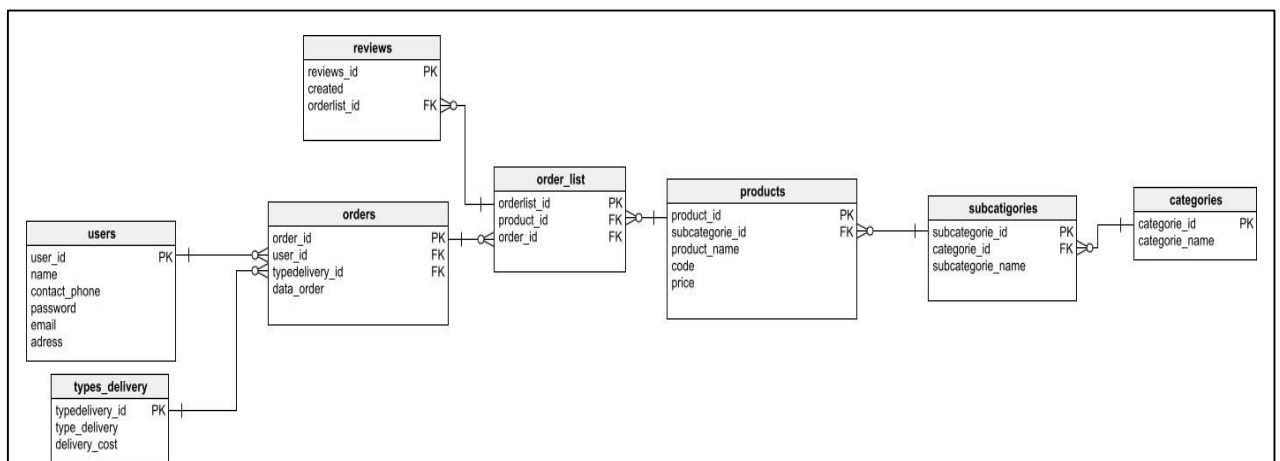


Рисунок 2 – Логическая модель базы данных

4. Физическое проектирование

В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведенным анализом предметной области и была проработана следующая физическая схема базы данных. Она представлена на следующем рисунке:

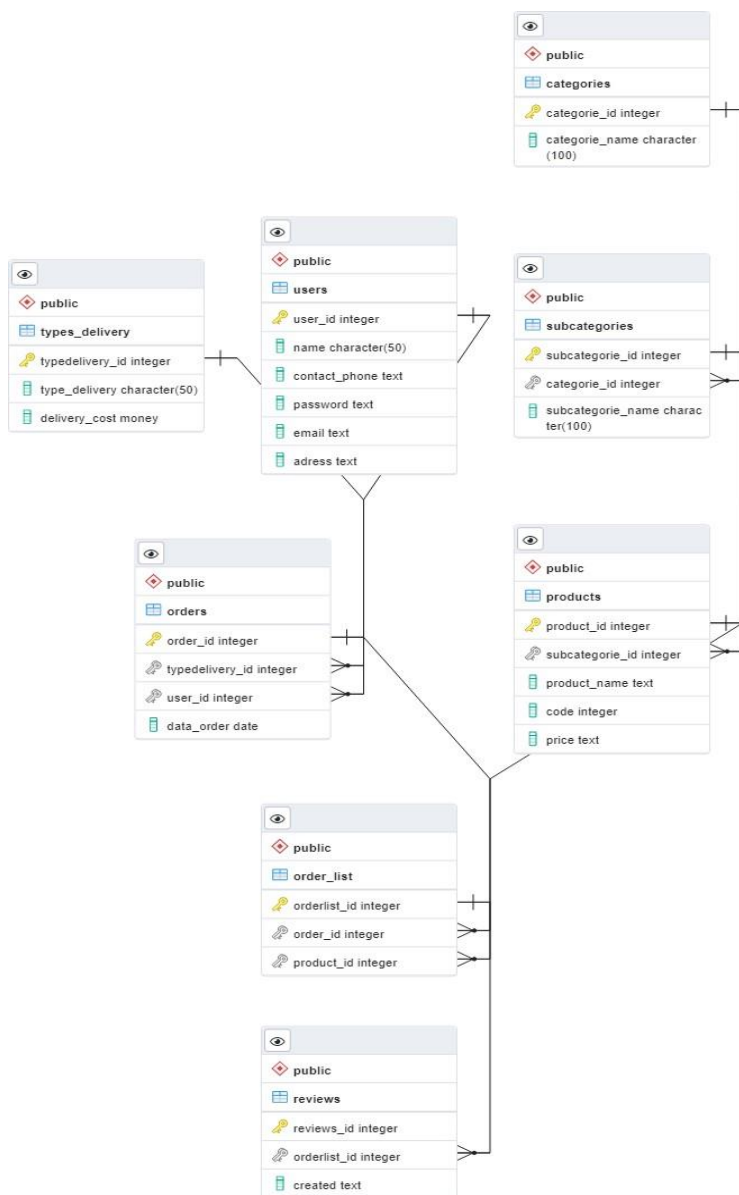


Рисунок 3 — физическая модель базы данных.

4.1 Создания таблиц.

Создание таблицы категории:

```
CREATE TABLE IF NOT EXISTS categories(  
  categorie_id serial PRIMARY KEY,  
  categorie_name text NOT NULL  
);
```

Создание таблицы подкатегории:

```
CREATE TABLE IF NOT EXISTS subcategories(  
  subcategorie_id serial PRIMARY KEY,  
  categorie_id integer NOT NULL REFERENCES categories ON DELETE CASCADE,  
  subcategorie_name text NOT NULL  
);
```

Создание таблицы товары:

```
CREATE TABLE IF NOT EXISTS products(  
  product_id serial PRIMARY KEY,  
  subcategorie_id integer NOT NULL REFERENCES subcategories ON DELETE CASCADE,  
  product_name text NOT NULL,  
  code integer NOT NULL,  
  price real NOT NULL  
);
```

Создание таблицы пользователи:

```
CREATE TABLE IF NOT EXISTS users(  
  user_id serial PRIMARY KEY,  
  name text NOT NULL,  
  contact_phone integer NOT NULL,  
  password text CHECK NOT NULL,  
  email text NOT NULL,  
  adress text NOT NULL  
);
```

Создание таблицы тип доставки:

```
CREATE TABLE IF NOT EXISTS types_delivery(  
  typedelivery_id serial PRIMARY KEY,  
  type_delivery text NOT NULL,  
  delivery_cost real NOT NULL  
);
```

Создание таблицы заказы:

```
CREATE TABLE IF NOT EXISTS orders(  
  order_id serial PRIMARY KEY,  
  typedelivery_id integer NOT NULL REFERENCES types_delivery ON DELETE CASCADE,  
  user_id integer NOT NULL REFERENCES users ON DELETE CASCADE,  
  data_order date NOT NU
```

```
);
```

Создание таблицы состав заказа:

```
CREATE TABLE IF NOT EXISTS order_list(  
    orderlist_id serial PRIMARY KEY,  
    order_id integer NOT NULL REFERENCES orders ON DELETE CASCADE,  
    product_id integer NOT NULL REFERENCES products ON DELETE CASCADE  
);
```

Создание таблицы отзывы:

```
CREATE TABLE IF NOT EXISTS reviews(  
    reviews_id serial PRIMARY KEY,  
    orderlist_id integer NOT NULL REFERENCES order_list ON DELETE CASCADE,  
    created text NOT NULL  
);
```

4.2. Заполнение базы данных

4.2.1. Подготовка данных

Для заполнения таблиц: категория, подкатегория и товары, на языке python был написан парсер, позволяющий собирать данные о товарах с интернет магазина iHerb. Была собрана информация об 35000 товаров, входящих в 100 подкатегорий, находящихся в 10 категориях. Все данные были записаны в csv файлы.

```
HOST = 'https://iherb.group/'  
URL = 'https://iherb.group/category/kosti-sustavy-i-hryashchi/'  
def get_html(url, params=""):  
    r = requests.get(url, headers=HEADERS, params=params)  
    return r  
def get_content(html):  
    soup = BeautifulSoup(html, 'html.parser')  
    items = soup.find_all('form', class_='js-add-to-cart')  
    product = []  
  
    for item in items:  
        product.append(  
            {  
                'Product-grid_name': item.find('div', class_='Product-grid_name').get_text(strip=True),  
                'nowrap price': item.find('div', class_='nowrap price').get_text(strip=True),  
                'Product-grid_action': item.find('div', class_='Product-grid_action').find('span').get('data-product')  
            }  
        )  
    return product  
def save_doc(items, path):  
    with open(path, 'w', newline="") as file:  
        writer = csv.writer(file, delimiter=';')  
        writer.writerow(['Название', 'Цена'])  
        for item in items:  
            writer.writerow([item['Product-grid_name'], item['nowrap price']])  
def parser():  
    PAGINATION = input('Укажи страницы ')
```



```

PAGINATION = int(PAGINATION.strip())
html = get_html(URL)
product = []
for page in range(1, PAGINATION + 1):
    print(f'Парсим страницу : {page}')
    html = get_html(URL, params={'page': page})
    product.extend(get_content(html.text))
with open('Кости, суставы и хрящи.csv', 'w', encoding='utf8', newline='') as f:
    thewriter = writer(f)
    header = ['Название', 'Цена', 'Артикул']
    thewriter.writerow(header)
    for item in product:
        info = [item['Product-grid_name'], item['nowrap price'], item['Product-grid_action']]
        thewriter.writerow(info)

```

Для заполнения таблицы: пользователь, в сети интернет были найдены списки имён, фамилий, названий городов и улиц, номера телефона и электронные почты. Для заполнения таких атрибутов, как дата заказа, пароль пользователя и других с помощью Python3 были использованы генераторы случайных данных.

4.2.2. Программа заполнения базы данных

Для заполнения базы данных был написан скрипт на языке программирования Python3 с использованием драйвера взаимодействия с СУБД PostgreSQL psycopg2. Общая схема работы скриптов заключается в следующем: С помощью драйвера psycopg2 осуществляется подключение к базе данных, внутри транзакции выполняется сформированный запрос, после чего подключение закрывается. Как видно, происходит обыкновенный INSERT запрос, который содержит данные, выбранные случайным образом. Во втором примере также осуществляется SELECT запрос к БД, чтобы на выходе получить относительно достоверные данные. В данном случае, заказы клиента осуществляются в городах, который он указывал в своих адресах

Заполнение таблицы категории:

```

categorie = ["Дом", "Здоровье", "Красота", "Питание для физической активности", "Питомцы",
"Пищевые добавки", "Продукты питания", "Средства для душа и ухода", "Товары для детей",
"Травы и натуральные средства"]

def insert_categorie():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("DELETE FROM categories")
    for i in range(len(categorie)):
        cur.execute("INSERT INTO categories(categorie_name) VALUES(%s)", (categorie[i],))
    conn.commit()

```

При заполнение таблицы категории происходят обыкновенные INSERT запросы, которые содержит данные находящиеся в заранее подготовленном массиве categorie.

Заполнение таблицы подкатегории:

```

def insert_subcategorie():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("SELECT categorie_id FROM categories")
    massiv = cur.fetchall()
    cur.execute("DELETE FROM subcategories")

```

```

for i in range(len(subcategorie)):
    for j in range(len(subcategorie[i])):
        cur.execute("INSERT INTO subcategories(categorie_id, subcategorie_name) VALUES(%s, %s)",
(massiv[i], subcategorie[i][j]))
    conn.commit()

```

При заполнении таблицы подкатегории осуществляется SELECT запрос к БД, чтобы извлечь находящиеся там категории, после чего они записываются в массив. После этого происходит INSERT запрос, в котором содержатся заранее подготовленные данные о подкатегориях и соответственно извлеченные из БД данные о категориях. Таким образом мы получаем достоверные данные о подкатегориях, входящих в определённые категории товаров.

Заполнение таблицы тип доставки:

```

delivery = [{"Курьерская доставка", 3000}, {"Срочная курьерская
доставка", 2500}, {"Постаматы", 2000}, {"Почта", 1500}]
def insert_delivery():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("DELETE FROM types_delivery")
    for i in range(len(delivery)):
        cur.execute("INSERT INTO types_delivery(type_delivery, delivery_cost) VALUES(%s, %s)",
(delivery[i][0], delivery[i][1]))
    conn.commit()

```

При заполнении таблицы тип доставки осуществляются INSERT запросы, в которых содержатся заранее подготовленные данные о типах доставок.

Заполнение таблицы пользователи:

```

def insert_users():
    user = read_users("пользователь.csv")
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("DELETE FROM users")
    for i in range(len(user)):
        cur.execute("INSERT INTO users(name, contact_phone, password, email, adress) VALUES(%s,
%s, %s, %s, %s)", (user[i]['Фамилия'] + " " + user[i]['Имя'] + " " + user[i]['Отчество'], user[i]['Номер
телефона'], user[i]['Пароль'], user[i]['Электронная почта'], user[i]['Город'] + " " + user[i]['Улица'] + " "
+ user[i]['Номер дома']))
    conn.commit()

def read_file_product(file_name):
    with open(file_name) as r_file:
        file_reader = csv.DictReader(r_file, delimiter=";")
        users = []
        for row in file_reader:
            users.append(row)
    return users

```

При заполнении таблицы пользователи осуществляются INSERT запросы, в которых содержатся данные извлеченные из подготовленного csv файла с помощью написанной функции read_file_product.

Заполнение таблицы товары:

```

def insert_product():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    directory = 'D:\BD_SQL\продукты'
    finish = []
    cur.execute("DELETE FROM products")
    for filename in os.listdir(directory):
        f = os.path.join(directory, filename)
        a = f
        adressa = []
        for i in os.listdir(a):
            k = 0
            adressa.append(os.path.join(a, i))
        finish.append(adressa)
    cur.execute("SELECT subcategorie_id FROM subcategories")
    massiv = cur.fetchall()
    print(massiv)
    l = 0
    for i in range(len(finish)):
        print(i)
        for adres in finish[i]:
            pr = read_file_product(f"{adres}")
            for j in range(len(pr)):
                cur.execute("INSERT INTO products(subcategorie_id, product_name, code, price)
VALUES(%s, %s, %s, %s)",(massiv[l], pr[j]['Название'], int(pr[j]['Артикул']), pr[j]['Цена']))
                l = l + 1
    conn.commit()

```

При заполнении таблицы товары осуществляется SELECT запрос к БД, чтобы извлечь находящиеся там подкатегории, после чего они записываются в массив. После этого происходят INSERT запросы, в которых содержатся заранее подготовленные в csv файлах данные о товарах и соответственно извлеченные из БД данные о подкатегориях. Таким образом мы получаем достоверные данных о товарах, входящих в определённые подкатегории.

Заполнение таблицы заказы:

```

def fill():
    nums = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("SELECT * FROM types_delivery")
    massiv1 = cur.fetchall()
    cur.execute("SELECT * FROM users")
    massiv2 = cur.fetchall()
    cur.execute("DELETE FROM orders")
    a = []
    b = []
    year = [2018, 2019, 2020, 2022]
    month = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
    day = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]
    for i in range(100):
        d = random.choice(massiv2)
        n = random.choice(nums)
        for j in range(n):
            a.append(d)
            b.append(random.choice(massiv1))

```

```

k = random.choice(year)
l = random.choice(month)
m = random.choice(day)
date1 = datetime.date(k, l, m)
c.append(date1)
for i in range(len(a)):
    cur.execute("INSERT INTO orders( typedelivery_id,user_id,data_order) VALUES(%s,%s,%s)",
(b[i][0],a[i][0],c[i]))
conn.commit()

```

При заполнении таблицы товары осуществляется SELECT запрос к БД, чтобы извлечь находящуюся там информацию о пользователях и типах доставки. Затем генерируются случайные даты заказов. После этого происходят INSERT запросы в которых содержатся случайно выбранные пользователи и типы доставок, находящихся в БД на момент заполнения данной таблицы. Таким образом мы получаем данных о заказах, содержащие id пользователей и id типов доставок находящихся в БД.

Заполнение таблицы состав заказа:

```

def fill_2():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("SELECT * FROM orders")
    massiv1 = cur.fetchall()
    cur.execute("SELECT * FROM products")
    massiv2 = cur.fetchall()
    nums = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
    idorder = []
    idprod = []
    cur.execute("DELETE FROM order_list")
    for i in range(len(massiv1)):
        n = random.choice(nums)
        for j in range(n):
            idorder.append(massiv1[i])
            idprod.append(random.choice(massiv2))
    for i in range(len(idorder)):
        cur.execute("INSERT INTO order_list(order_id,product_id) VALUES(%s,%s)", (idorder[i][0],
idprod[i][0]))
    conn.commit()

```

При заполнении таблицы товары осуществляется SELECT запрос к БД, чтобы извлечь находящуюся там информацию о заказах и товарах. После этого происходят INSERT запросы в которых содержатся случайно выбранные заказы и товары извлеченные из БД на момент создания данной таблицы. Таким образом мы получаем данные о составе заказа, содержащего id заказа и id пользователя находящихся в БД.

Заполнение таблицы отзывы:

```

def fill_3():
    conn = psycopg2.connect(dbname="iHerb", user="postgres", password="dima78917")
    cur = conn.cursor()
    cur.execute("SELECT * FROM order_list")
    massiv1 = cur.fetchall()
    mas=[]
    cur.execute("DELETE FROM reviews")

```

```

for i in range(len(massiv1)):
    mas.append(massiv1[i][0])
#print(mas)
NARECH = ['Круто.', 'Хорошо.', 'Нормально.', 'Отлично.', 'Супер.', 'Классно.', 'Неплохо.',
'Идеально.', 'Красота. ',
    'Прекрасно. ', 'Похвально. ', 'Щедро. ', 'Качественно. ', 'Клево. ', 'Превосходно. ', 'Отменно. ',
    'Годится. ', 'Доброкачественно. ', 'Тип-топ. ', 'На хорошем уровне. ', 'Распрекрасно. ', 'Блеск.
',
    'Отл. ', 'Достаточно. ', 'Все окей. ', 'В порядке. ']
SUB = ['Товар ', 'Продукт ', 'Продукция ', 'Вещь ', 'Вещица ', 'Изделие ', 'Заказ ']
NARE = ['супер. ', 'высший класс. ', 'не очень. ', 'отпад! ']
VERB = ['Понравилось', 'Обрадовало', 'Запало в душу', 'Но не оправдало ожиданий', 'Но не это
хотели',
    'Но не очень понравилось']
ЕМОJI = ['=|', '=)', '!!!', ':|', 'o_o', '!', '^_^', '...', ':', '']
a = []
b = []
n = 100
a.append(random.sample(mas, n))
for i in range(n):
b.append(random.choice(NARECH)+random.choice(SUB)+random.choice(NARE)+random.choice(VER
B)+random.choice(ЕМОJI))
for i in range(n):
    cur.execute("INSERT INTO reviews(orderlist_id,created) VALUES(%s,%s)", (a[0][i], b[i]))
    conn.commit()

```

При заполнении таблицы отзывы осуществляется SELECT запрос к БД, чтобы извлечь находящуюся там информацию о составе заказа. После этого происходят INSERT запросы в которых содержатся случайно выбранные элементы из таблицы состав заказа и случайно выбранные слова из заранее подготовленных массивов с различным текстом. Таким образом мы получаем данные об отзывах.

4.3 Результаты заполнения

Результат заполнения таблицы категория:

	categorie_id [PK] integer	categorie_name character (100)
1	661	Дом
2	662	Здоровье
3	663	Красота
4	664	Питание для физической активности
5	665	Питомцы
6	666	Пищевые добавки
7	667	Продукты питания
8	668	Средства для душа и ухода
9	669	Товары для детей
10	670	Травы и натуральные средства

Результат заполнения таблицы подкатегория:

	subcategorie_id [PK] integer	categorie_id integer	subcategorie_name character (100)
1	2829	661	Домашняя утварь
2	2830	661	Уборка
3	2831	661	Хозяйственные товары
4	2832	662	Глаза и зрение
5	2833	662	Детоксикация и очищение
6	2834	662	Иммунная система
7	2835	662	Кровеносная система
8	2836	662	Мочевой пузырь
9	2837	662	Обезболивание
10	2838	662	Органы дыхания
11	2839	662	Простуда и грипп
12	2840	662	Простуда, кашель и грипп

Результат заполнения таблицы тип доставки:

	typedelivery_id [PK] integer	type_delivery character (50)	delivery_cost real
1	109	Курьерская доставка	3000
2	110	Срочная курьерская доставка	2500
3	111	Постаматы	2000
4	112	Почта	1500

Результат заполнения таблицы товары:

	product_id [PK] integer	subcategorie_id integer	product_name text	code integer	price text
1	731675	2829	PEAKfresh USA, многоразовые пакеты с затычками для хранения продуктов, 10 шт.	6989	610
2	731676	2829	ALLMAX Nutrition, герметичный шейкер, бутылка без БФА с миксером Vortex, 700 мл (25 унций)	17546	940
3	731677	2829	Think, Thinksport, герметичная спортивная бутылка, серебро, 25 унций (750 мл)	17542	2790
4	731678	2829	Blender Bottle, Classic With Loop, классический шейкер с петелькой, черный 600 мл (20 унций)	16549	1330
5	731679	2829	Blender Bottle, Classic, шейкер, черный, 828 мл (28 унций)	18020	1520
6	731680	2829	Stasher, Многоразовый силиконовый контейнер для еды, удобный размер для бутербродов, средний, прозрачный, 450 мл (15 жидк. унций)	4750	1670
7	731681	2829	Think, Thinksport, изолированная бутылка для спорта, мятный зеленый, 17 унций (500 мл)	13526	2700
8	731682	2829	Blender Bottle, Classic With Loop, классический шейкер с петелькой, сливовый, 600 мл (20 унций)	16553	1330
9	731683	2829	Blender Bottle, Classic with Loop, сливовый, 828 мл (28 унций)	18017	1490
10	731684	2829	Blender Bottle, Radian, из изолированной нержавеющей стали, матовый черный, 770 мл (26 унций)	17929	3280
11	731685	2829	Think, Thinksport, Insulated Sports Bottle, Dark Pink, 25 oz (750ml)	17539	3010
12	731686	2829	Blender Bottle, ProStak, черный, 650 мл (22 унции)	17066	1950
13	731687	2829	Blender Bottle, Classic With Loop, классический шейкер с петелькой, серый, 600 мл (20 унций)	16551	1330
14	731688	2829	Blender Bottle, Classic with Loop, серая галька, 828 мл (28 унций)	18015	1490
15	731689	2829	Blender Bottle, ProStak, черный, 651 мл (22 унции)	17057	2060
16	731690	2829	Blender Bottle, Classic With Loop, классический шейкер с петелькой, океанический голубой, 600 мл (20 унций)	16552	1330
17	731691	2829	Blender Bottle, Classic with Loop, классический шейкер с петелькой, океанический голубой, 828 мл (28 унций)	18010	1530
18	731692	2829	Think, Thinksport, герметичная спортивная емкость, зеленая мята, 25 унций (750 мл)	17540	3160
19	731693	2829	Think, Thinksport, термоизолированная спортивная бутылка, 750 мл (25 унций)	17538	3020

Результат заполнения таблицы состав заказа:

	orderlist_id [PK] integer	order_id integer	product_id integer
1	251570	28613	754536
2	251571	28613	741034
3	251572	28613	762323
4	251573	28613	744962
5	251574	28613	750810
6	251575	28613	764782
7	251576	28613	745028
8	251577	28613	755510
9	251578	28613	754377
10	251579	28613	753053
11	251580	28613	744891
12	251581	28613	735416
13	251582	28614	763190
14	251583	28614	760028
15	251584	28614	748945
16	251585	28614	742978
17	251586	28614	763251
18	251587	28614	748935
19	251588	28615	761087

Результат заполнения таблицы заказ:

	order_id [PK] integer	typedelivery_id integer	user_id integer	data_order date
1	28613	111	29918	2018-04-13
2	28614	110	29918	2022-01-12
3	28615	110	29918	2020-06-22
4	28616	112	29918	2019-11-22
5	28617	109	29918	2018-09-28
6	28618	109	29918	2020-11-19
7	28619	112	29918	2020-02-02
8	28620	111	29918	2019-02-05
9	28621	112	29918	2019-03-05
10	28622	110	29918	2018-02-22
11	28623	110	29918	2022-06-02
12	28624	110	29918	2019-04-02
13	28625	112	29918	2020-12-07
14	28626	110	29474	2022-03-10
15	28627	109	29474	2022-12-13
16	28628	111	29474	2019-05-03
17	28629	111	29474	2018-08-20
18	28630	111	29474	2019-03-06
19	28631	109	29474	2020-08-25
20	28632	111	29474	2020-01-24

Результат заполнения таблицы отзывы:

	reviews_id [PK] integer	orderlist_id integer	created text
1	1719	257537	Отлично.Вещица не очень. Запало в душу=
2	1720	259780	Отменно. Изделие отпад! Обрадовалоо_0
3	1721	254244	Качественно. Продукция супер. Обрадовалоо_0
4	1722	254388	Блеск. Изделие отпад! Обрадовало=)
5	1723	256139	Красота. Товар высший класс. Но не очень понравилось^_^
6	1724	254491	На хорошем уровне. Продукция отпад! Обрадовало!
7	1725	255794	Круто.Вещица не очень. Но не это хотели!!!
8	1726	257094	Качественно. Продукт супер. Запало в душу=)
9	1727	256474	Клево. Изделие высший класс. Но не это хотели^_^
10	1728	257729	Отл. Продукция не очень. Запало в душу!!!
11	1729	256979	Доброкачественно. Заказ не очень. Но не оправдало ожиданий:
12	1730	256587	Отлично.Продукция не очень. Но не очень понравилось!

5. Выполнение запросов

1. Количество товаров в заказе

```

WITH products_orderlist AS (
    SELECT DISTINCT order_id, count(*) OVER (PARTITION BY order_id)
    FROM order_list
    ORDER BY count DESC
), product_orders AS(
    SELECT orders.order_id, user_id, count
    FROM orders
    INNER JOIN products_orderlist ON products_orderlist.order_id = orders.order_id
)
SELECT product_orders.order_id, users.user_id, users.name, count
FROM users
INNER JOIN product_orders ON product_orders.user_id = users.user_id;

```


	order_id integer	user_id integer	name text	count bigint
9	4117	3749	Грибков Константин Конста...	18
10	1936	3529	Головченко Альбина Тимоф...	18
11	4202	3753	Апевалова Ирина Юрьевна	18
12	7373	3310	Ермолин Петр Акимович	18
13	8436	3400	Колтышева Лидия Валентин...	18
14	110	3370	Вятт Константин Тимофеевич	18
15	4370	3103	Кононов Арсений Васильевич	18
16	8340	3795	Савицкий Федот Себастьяно...	18
17	4657	3433	Славакова Альбина Григорь...	18
18	1550	3385	Петраков Гавриил Юрьевич	18
19	2549	3323	Мятлев Василий Фадеевич	18

2. Самый дорогой заказ в определенный период (за 2020 год)

```

WITH data_order AS (
    SELECT DISTINCT order_id, data_order
    FROM orders
    WHERE EXTRACT(YEAR FROM orders.data_order) = 2020
), orders_comp AS(
    SELECT order_list.order_id, product_id, orderlist_id
    FROM order_list
    INNER JOIN data_order ON data_order.order_id = order_list.order_id
), product_quant AS (
    SELECT order_id, product_id
    FROM orders_comp
    GROUP BY order_id, product_id
    ORDER BY product_id ASC
), product_total AS (
    SELECT order_id, SUM(products.price) as total_price
    FROM products
    INNER JOIN product_quant
    ON products.product_id = product_quant.product_id
    GROUP BY order_id
)
SELECT MAX(total_price)
FROM product_total;

```

	max real
1	70600

3. Посчитать среднее значение стоимости заказов

```


WITH product_quant AS (
    SELECT order_id, product_id
    FROM order_list
    GROUP BY order_id, product_id
    ORDER BY product_id ASC

```

```

), product_total AS (
    SELECT order_id, SUM(products.price) as total_price
    FROM products
    INNER JOIN product_quant
    ON products.product_id = product_quant.product_id
    GROUP BY order_id
)
SELECT AVG(total_price) AS average_price FROM product_total;

```



	average_price double precision 
1	22292.5560964027

4. Вывести самый дорогой и самый дешевый заказ

```

WITH product_quant AS (
    SELECT order_id, product_id
    FROM order_list
    GROUP BY order_id, product_id
    ORDER BY product_id ASC
), product_total AS (
    SELECT order_id, SUM(products.price) as total_price
    FROM products
    INNER JOIN product_quant
    ON products.product_id = product_quant.product_id
    GROUP BY order_id
)
SELECT MAX(total_price) AS max_price, MIN(total_price) AS min_price FROM product_total;

```

	max_price real 	min_price real 
1	74270	130

5. Топ 3 самых заказываемых товара:

```

WITH top_3_products_ids AS (
    SELECT product_id, COUNT (product_id)
    FROM order_list
    GROUP BY product_id
    ORDER BY COUNT (product_id) DESC LIMIT 3
)
SELECT products.product_name, products.code, top_3_products_ids.count
FROM products
INNER JOIN top_3_products_ids
ON products.product_id = top_3_products_ids.product_id
ORDER BY count DESC;

```

	product_name text	code integer	count bigint
1	Enzymatic Therapy, Очищение вс...	18473	11
2	Tasty Bite, Органическая индийс...	6693	11
3	Physician's Choice, Бузина с эхи...	27325	11

6. Получить товары определенной подкатегории = 409

```

WITH p_w AS (
    SELECT products.product_id, subcategories.category_id
    FROM products
    INNER JOIN subcategories
    ON products.subcategorie_id = subcategories.subcategorie_id
    WHERE subcategories.subcategorie_id = 409
)
SELECT products.product_id, subcategorie_id, products.product_name
FROM products
INNER JOIN p_w
ON products.product_id = p_w. product_id;

```

	product_id [PK] integer	subcategorie_id integer	product_name text
1	74730	409	petnc NATURAL CARE, средство для здоровья таза и суставов, только для собак, со вкусом печени, 90 мягких ж...
2	74731	409	NaturVet, ArthriSoothe-GOLD, профессиональный уход, уровень 3, 180 мягких подушечек, 15.2 унций (432 г)
3	74732	409	NaturVet, Универсальное средство, поддержка 4 в 1, для собак, 120 жевательных таблеток, 480 г (16,9 унции)
4	74733	409	Nordic Naturals, Omega-3 Pet, для собак, 180 капсул
5	74734	409	Charlie & Frank, Рыбий жир с омега-3 для животных, для кошек и собак, 100 мл (3,3 жидк. унции)
6	74735	409	Natural Dog Company, Hip & Joint, для всех возрастов, куриная печень и куркума, 90 жевательных таблеток, 284 ...
7	74736	409	NaturVet, ArthriSoothe-GOLD, улучшенный уход, уровень 3, 120 жевательных таблеток, 21 унц. (600 г)
8	74737	409	ION Biome, поддержка кишечника домашних питомцев, для собак и кошек, 437 мл (16 жидк. унций)
9	74738	409	Nordic Naturals, Омега-3 для домашних животных, 237 мл (8 жидких унций)
10	74739	409	Charlie & Frank, Улучшенная формула для здоровья суставов животных с омега-кислотами, для кошек и собак, ...
11	74740	409	NaturVet, Glucosamine DS Plus, умеренный уход, уровень 2, 240 жевательных таблеток, 576 г (1 фунт 4 унции)
12	74741	409	NaturVet, VitaPet Senior, ежедневные витамины и глюкозамин для собак, 120 жевательных таблеток, 360 г (12,6...

7. Найти клиентов у которых больше трех заказов:

```

WITH us_order_1 AS (
    SELECT DISTINCT A.user_id AS us_id_A, A.order_id AS or_id_A, B.order_id AS or_id_B
    FROM orders AS A, orders AS B
    WHERE A.user_id = B.user_id AND A.order_id <> B.order_id
    ORDER BY A.user_id ASC
), us_order_2 AS (
    SELECT DISTINCT D.user_id, D.order_id
    FROM us_order_1 AS C
    INNER JOIN orders AS D
    ON D.user_id = C.us_id_A AND D.order_id <> C.or_id_A AND D.order_id <> C.or_id_B
    ORDER BY D.user_id ASC
), users_ids AS (

```

```

SELECT DISTINCT orders.user_id
FROM orders
INNER JOIN us_order_2
ON orders.user_id = us_order_2.user_id AND orders.order_id <> us_order_2.order_id
)

```

```

SELECT DISTINCT users.*
FROM users
INNER JOIN users_ids
ON users.user_id = users_ids.user_id
ORDER BY users.user_id ASC;

```

	user_id [PK] integer	name text	contact_phone text	password text	email text	adress text
1	3001	Маховицкая Клара Сергеевна	+7 (941) 576-41-62	291575b76	klara.mahovickaya@ram...	г. Сыктывкар Новоселов ул...
2	3003	Морякова Наталия Михайловна	+7 (918) 293-92-24	e972189e1	nataliya95@gmail.com	г. Красноярск Советский пе...
3	3004	Лелух Арсений Прокопьевич	+7 (952) 497-60-29	e62dd61e1	arseniy1961@gmail.com	г. Бийск Песчаная ул. 11
4	3005	Андронов Павел Егорович	+7 (999) 516-46-43	2b20ba64c	pavel.andronov@outlook...	г. Нижний Новгород Полево...
5	3007	Калошина Марфа Ростиславовна	+7 (970) 767-16-96	62b41409d	marfa75@yandex.ru	г. Королёв Коммунистическ...
6	3008	Кизатов Федот Денисович	+7 (908) 166-55-28	48ed646e9	fedot8920@mail.ru	г. Долгопрудный Южная ул. ...
7	3009	Ругов Прохор Саввеевич	+7 (921) 442-51-87	35ab727c9	prohor94@gmail.com	г. Брянск Зеленая ул. 24
8	3011	Полков Алексей Нифонтович	+7 (990) 911-41-86	b00e4d58a	aleksey09031961@yand...	г. Волжский Речной пер. 21
9	3015	Забабурина Лариса Арсеньевна	+7 (998) 371-97-68	b40986e00	larisa.zababurina@gmail...	г. Бердск Новая ул. 8
10	3018	Дятлов Кирилл Георгиевич	+7 (971) 827-15-77	5e8ea6992	kirill.dyatlov@rambler.ru	г. Энгельс Березовая ул. 10

8. Вывести последний заказ содержащий определенный продукт

```

WITH prod_id AS (
    SELECT product_id
    FROM products
    WHERE product_name = 'Blender Bottle, ProStak, черный, 651 мл (22 унции)'
), orders_with_prod AS (
    SELECT order_id
    FROM order_list AS ord_1
    INNER JOIN prod_id
    ON ord_1.product_id = prod_id.product_id
)
SELECT orders.*
FROM orders
INNER JOIN orders_with_prod AS owp
ON orders.order_id = owp.order_id
ORDER BY data_order DESC LIMIT 1;

```

	order_id [PK] integer	typedelivery_id integer	user_id integer	data_order date
1	483	15	3861	2019-06-28

9. Вывести список заказов, на которые определённый пользователь (3021) потратил более (1500)

```

WITH user_orders AS (
    SELECT order_id
    FROM orders
    WHERE user_id = 3021
), orders_comp AS (
    SELECT user_orders.order_id, product_id
    FROM order_list
    INNER JOIN user_orders
    ON order_list.order_id = user_orders.order_id
    ORDER BY product_id ASC
), product_quant AS (

```

```

SELECT order_id, product_id, count(product_id) AS quantity
FROM orders_comp
GROUP BY order_id, product_id
ORDER BY product_id ASC
), product_total AS (
SELECT order_id, SUM(products.price*product_quant.quantity) as total_price
FROM products
INNER JOIN product_quant
ON products.product_id = product_quant.product_id
GROUP BY order_id
)
SELECT *
FROM product_total
WHERE CAST(total_price AS NUMERIC) >= 1500;

```

	order_id [PK] integer	total_price double precision
1	2361	23260
2	2362	5770
3	2363	1850
4	2364	17020
5	2366	43450
6	2367	65150
7	7682	19640
8	7683	23500

10. Найти позиции ассортимента с ценами <500 . Фильтрация ассортимента в приложении по ценам.

```


WITH prod AS(
SELECT product_name, price
FROM products
WHERE CAST(price AS NUMERIC) < 500
)
SELECT *
FROM prod
ORDER BY CAST(price AS NUMERIC);

```

	product_name text	price real
1	iHerb Goods, Сумка для бакалеи, очень большая	20
2	Idealove, маска для кожи с суперфудами, лайм, 1 тканевая маска, 20 мл (0,68 жидк. унции)	130
3	Idealove, Superfood Skin Savior, от любви и меда, 1 тканевая маска, 20 мл (0,68 жидк. Унции)	130
4	Idealove, Superfood Skin Savior, от любви и меда, 1 тканевая маска, 20 мл (0,68 жидк. Унции)	130
5	Radiant Seoul, осветляющая тканевая маска, 1 шт., 25 мл (0,85 унции)	130
6	Idealove, Superfood Skin Savior, Acai You Checking Me Out, 1 тканевая маска, 20 мл (0,68 жидк. Унции)	130
7	California Gold Nutrition, пребиотическая клетчатка, 3 пакетика, по 6 г (0,21 унции)	130
8	Mild By Nature, увлажняющий кондиционер с ягодами асаи, компактный размер, 63 мл (2,1 жидк. унции)	130
9	Radiant Seoul, тканевая маска для объема и гладкости кожи, 1 шт., 25 мл (0,85 унции)	130
10	Mild By Nature, увлажняющий шампунь с ягодами асаи, компактный размер, 63 мл (2,10 жидк. унции)	130
11	Radiant Seoul, тканевая маска с древесным углем для восстановления баланса, 1 шт., 25 мл (0,85 унции)	130
12	Radiant Seoul, тканевая маска с древесным углем для восстановления баланса, 1 шт., 25 мл (0,85 унции)	130
13	Idealove, Superfood Skin Savior, Pretty as a Peach, 1 тканевая маска, 20 мл (0,68 жидк. Унции)	130
14	Idealove, Superfood Skin Savior, Almond Love with You, 1 тканевая маска, 20 мл (0,68 жидк. Унции)	130




11. Вычислить суммарную выручку с продаж за 2020 год. Для ведения бух. учёта.

```
WITH orders_in_2020 AS (
    SELECT product_id
    FROM order_list
    INNER JOIN orders
    ON order_list.order_id = orders.order_id
    WHERE CAST(orders.data_order AS TEXT) LIKE '2020%' ORDER BY product_id ASC
), products_quant AS (
    SELECT product_id, count(product_id) AS quantity
    FROM orders_in_2020
    GROUP BY product_id
    ORDER BY product_id ASC
), products_total AS (
    SELECT SUM(products.price*products_quant.quantity) total_price
    FROM products
    INNER JOIN products_quant ON products.product_id = products_quant.product_id
)
SELECT SUM(total_price) total FROM products_total;
```

	total double precision 
1	48228100

12. Найти количество товара в каждой подкатегории

```
WITH products_subcat AS (
    SELECT DISTINCT subcategorie_id, count(*) OVER (PARTITION BY subcategorie_id)
    FROM products
    ORDER BY count DESC
)
SELECT subcategories.subcategorie_id, subcategorie_name, count
FROM subcategories
INNER JOIN products_subcat ON products_subcat.subcategorie_id = subcategories.subcategorie_id;
```

	subcategorie_id [PK] integer 	subcategorie_name text 	count bigint 
1	405	Домашняя утварь	272
2	406	Уборка	1620
3	407	Хозяйственные то...	4540
4	408	Глаза и зрение	1801
5	409	Детоксикация и о...	411
6	410	Иммунная система	11069
7	411	Кровеносная сист...	3138
8	412	Мочевой пузырь	5805
9	413	Обезболивание	1906
10	414	Органы дыхания	2686

13. Список всех заказов за декабрь 2020 года

```
SELECT orders.order_id, data_order
```

```
FROM orders
WHERE (EXTRACT(MONTH FROM orders.data_order) = 12) AND (EXTRACT(YEAR FROM
orders.data_order) = 2020)
ORDER BY orders.data_order;
```

	order_id [PK] integer	data_order date
1	5030	2020-12-01
2	4989	2020-12-01
3	1099	2020-12-01
4	4463	2020-12-01
5	8063	2020-12-02
6	917	2020-12-02
7	4983	2020-12-02
8	2472	2020-12-02
9	1735	2020-12-02
10	8533	2020-12-02

14. Какие отзывы оставлял пользователь 3370

```
WITH user_orders AS (
    SELECT order_id
    FROM orders
    WHERE user_id = 3370
), orders_comp AS (
    SELECT user_orders.order_id, orderlist_id
    FROM order_list
    INNER JOIN user_orders
    ON order_list.order_id = user_orders.order_id
    ORDER BY orderlist_id ASC
), product_quant AS (
    SELECT created
    FROM reviews
    INNER JOIN orders_comp
    ON orders_comp.orderlist_id=reviews.orderlist_id
)
SELECT *
FROM product_quant;
```

	created text
1	Годится. Вещь супер. Но не это хотели...
2	Неплохо. Вещь супер. Но не очень пон...

15. Пользователи, которые не совершали заказы

```
SELECT user_id,name
FROM users
```


LEFT JOIN orders USING(user_id)
 WHERE order_id IS NULL
 Order by name;

	user_id [PK] integer	name text
1	3045	Абалышев Максим Валентинович
2	3349	Авандеев Макар Иннокентиевич
3	3157	Азарова Пелагея Саввановна
4	3858	Акчурин Кирилл Герасимович
5	3053	Андрианова Марьяна Кузьминовна
6	3136	Анисимова Лидия Феоктистовна
7	3977	Анохина Ася Никифоровна
8	3835	Антипина Мария Севастьяновна
9	3660	Анюкова Алена Якововна
10	3773	Аристархова Ксения Евгеньевна
11	3150	Аристов Афанасий Юлианович
12	3853	Асланова Маргарита Михайловна

16. пользователей и адреса которые совершали заказ и города Йошкар - Ола

```
WITH extract_exp_info AS (
  SELECT users.user_id, adress
  FROM users
  INNER JOIN orders ON users.user_id = orders.user_id
  WHERE adress LIKE '%Йошкар-Ола%'
)
SELECT DISTINCT user_id,adress FROM extract_exp_info;
```

	user_id [PK] integer	adress text
1	3667	г. Йошкар-Ола Заречная ул. 24
2	3938	г. Йошкар-Ола Дружбы ул. 8
3	3370	г. Йошкар-Ола Социалистическ...

17. Сколько заказов сделал пользователь 3761

```
SELECT count(*) FROM orders
WHERE users_id = 3761;
```

	count bigint
1	21

18. Сколько раз заказывали каждым типом доставки

```
WITH orders_delivery AS (
  SELECT DISTINCT typedelivery_id,count(*) OVER (PARTITION BY typedelivery_id)
  FROM orders
  ORDER BY count DESC
```



```
)
SELECT types_delivery.typedelivery_id, type_delivery, count
FROM types_delivery
INNER JOIN orders_delivery ON orders_delivery.typedelivery_id = types_delivery.typedelivery_id;
```

	typedelivery_id [PK] integer	type_delivery text	count bigint
1	13	Курьерская доставка	2259
2	15	Постаматы	2246
3	14	Срочная курьерская доставка	2191
4	16	Почта	2174

19. Какой тип доставки был популярен в 2020

```
WITH orders_in_2020 AS (
    SELECT typedelivery_id, order_id
    FROM orders
    WHERE EXTRACT(YEAR FROM orders.data_order) = 2020
    ORDER BY typedelivery_id ASC
), orders_delivery AS (
    SELECT DISTINCT typedelivery_id, count(*) OVER (PARTITION BY typedelivery_id
    FROM orders_in_2020
    ORDER BY count DESC
), orders_deliverytype AS (
    SELECT types_delivery.typedelivery_id, type_delivery, count
    FROM types_delivery
    INNER JOIN orders_delivery
    ON orders_delivery.typedelivery_id = types_delivery.typedelivery_id
    ORDER BY count DESC LIMIT 1
)
SELECT *
FROM orders_deliverytype;
```

	typedelivery_id [PK] integer	type_delivery text	count bigint
1	15	Постаматы	604

20. В зависимости от суммы заказа вывести статус бонусов

```
WITH product_quant AS (SELECT order_id, product_id
FROM order_list
GROUP BY order_id, product_id
ORDER BY product_id ASC
),
total AS (
    SELECT order_id, SUM(CAST(products.price AS integer)) as total_price
    FROM products
    INNER JOIN product_quant
    ON products.product_id = product_quant.product_id
    GROUP BY order_id
)
SELECT order_id, total_price,
CASE WHEN total_price > 40000 THEN 'Бонус 5 звёзд'
      WHEN total_price > 30000 THEN 'Бонус 4 звезды'
```

```

WHEN total_price > 20000 THEN 'Бонус 3 звезды'
      WHEN total_price > 10000 THEN 'Бонус 2 звезды'
      ELSE 'Бонус 1 звезда'
END
FROM total;

```

	order_id integer	total_price bigint	case text
1	29411	35850	Бонус 4 звезды
2	29113	35820	Бонус 4 звезды
3	29329	12270	Бонус 2 звезды
4	29465	25150	Бонус 3 звезды
5	28723	29950	Бонус 3 звезды
6	28869	19020	Бонус 2 звезды
7	28970	31230	Бонус 4 звезды
8	29501	11950	Бонус 2 звезды
9	29474	6880	Бонус 1 звезда
10	29320	30280	Бонус 4 звезды
11	29084	23930	Бонус 3 звезды
12	29250	16630	Бонус 2 звезды
13	28939	45020	Бонус 5 звёзд
14	29046	2570	Бонус 1 звезда