# Задание 1

Книжный онлайн-магазин

Фактическая сущность:

- **Sale (продажа)**

Поля:

- sale_id
- buyer_id
- delivery_service_id
- warehouse_id
- total
- address
- DTTM

Измерения:

- **Book (книга)**

Поля:

- book_id
- publishing_id
- name
- author

- **Genre (жанр)**

Поля:

- genre_id
- name

- **Publishing (издательство)**

Поля:

- publishing_id
- name
- address
- city

- **Warehouse (склад)**

Поля:

- warehouse_id
- city
- address
- phone_number
- capacity
- current_stock

- **Delivery_service (служба доставки)**

Поля:

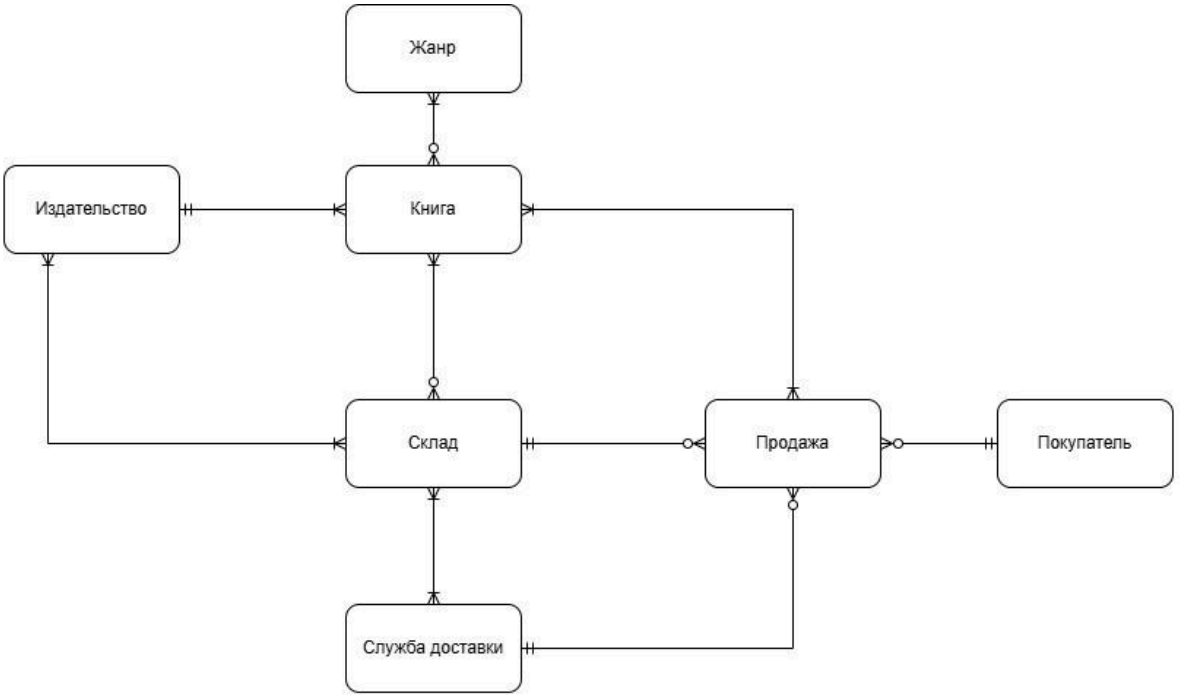
- delivery_service_id
- name

- **Buyer (покупатель)**

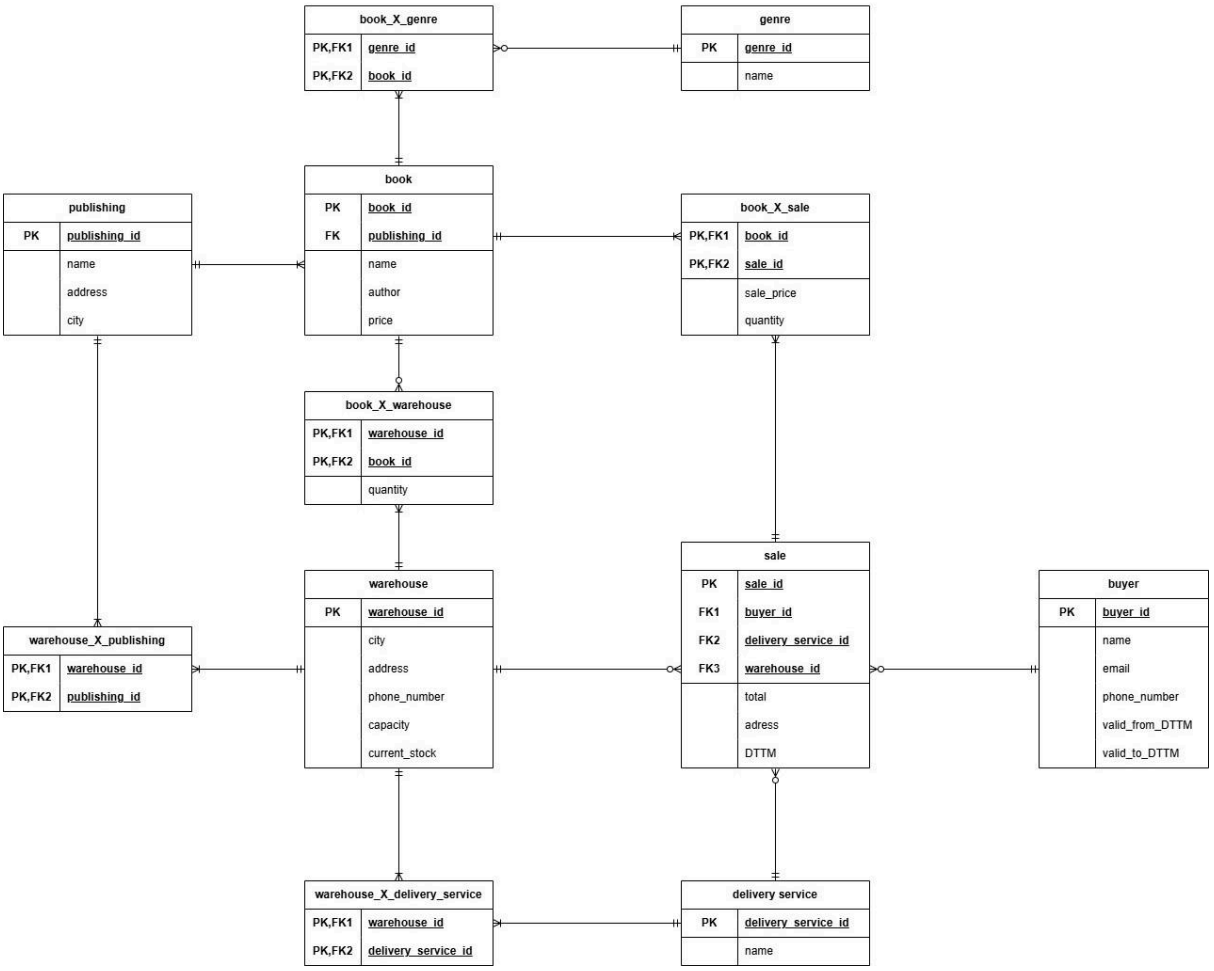
Поля:

- buyer_id
- name
- email
- phone_number
- valid_from_DTTM
- valid_to_DTTM

## Задание 2(а)



# Задание 2(b)




- Для базы данных была выбрана 3НФ, так как база данных содержит большое количество взаимосвязей таблиц, и это поможет избежать избыточности и дублирования данных. Важно, чтобы информация о покупателях, продажах и запасах книг на складе не пересекалась и оставалась независимой.

  Например, было решено выделить отдельную таблицу **publishing** для издательства вместо того, чтобы прописывать его атрибутом к каждой книге, так как это позволяет смотреть на издательства в контексте поставок на склад, а также не дублировать информацию о книге для каждой её печатной версии в случае, если книга издается в нескольких издательствах.

- Версионной является таблица **buye**r (SCD2)

# Задание 3

```sql
CREATE DATABASE bookstore_db;

CREATE SCHEMA bookstore;

-- Table Publishing (издательство)
CREATE TABLE bookstore.Publishing (
    publishing_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(255) NOT NULL
);

-- Table Warehouse (склад)
CREATE TABLE bookstore.Warehouse (
    warehouse_id SERIAL PRIMARY KEY,
    city VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    capacity INTEGER NOT NULL,
    current_stock INTEGER NOT NULL,
    CONSTRAINT check_current_stock_capacity CHECK (current_stock >= 0 AND
current_stock <= capacity)
);

-- Table Delivery_service (служба доставки)
CREATE TABLE bookstore.Delivery_service (
    delivery_service_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE
);
```

```sql
-- Table Buyer (покупатель)
CREATE TABLE bookstore.Buyer (
    buyer_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255),
    phone_number VARCHAR(20) NOT NULL,
    valid_from_DTTM TIMESTAMP NOT NULL,
    valid_to_DTTM TIMESTAMP,
    CONSTRAINT check_DTTM_validity CHECK (valid_to_DTTM IS NULL OR
valid_to_DTTM > valid_from_DTTM),
    CONSTRAINT check_email_format CHECK (email IS NULL OR email LIKE
'%@%.%')
);

-- Table Genre (жанр)
CREATE TABLE bookstore.Genre (
    genre_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE
);

-- Table Book (книга)
CREATE TABLE bookstore.Book (
    book_id SERIAL PRIMARY KEY,
    publishing_id INTEGER NOT NULL,
    name VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    price NUMERIC(15, 2) NOT NULL CHECK (price >= 0),
    CONSTRAINT FK_BookPublishing FOREIGN KEY (publishing_id) REFERENCES
bookstore.Publishing(publishing_id)
);
```

```sql
-- Table Sale (продажа)
CREATE TABLE bookstore.Sale (
    sale_id SERIAL PRIMARY KEY,
    buyer_id INTEGER NOT NULL,
    delivery_service_id INTEGER NOT NULL,
    warehouse_id INTEGER NOT NULL,
    address VARCHAR(255) NOT NULL,
    DTTM TIMESTAMP NOT NULL,
    CONSTRAINT FK_SaleBuyer FOREIGN KEY (buyer_id) REFERENCES
bookstore.Buyer(buyer_id),
    CONSTRAINT FK_SaleWarehouse FOREIGN KEY (warehouse_id) REFERENCES
bookstore.Warehouse(warehouse_id),
    CONSTRAINT FK_SaleDeliveryService FOREIGN KEY (delivery_service_id)
REFERENCES bookstore.Delivery_service(delivery_service_id)
);

-- Table Book_x_Warehouse (книга-склад)
CREATE TABLE bookstore.Book_x_Warehouse (
    warehouse_id INTEGER NOT NULL,
    book_id INTEGER NOT NULL,
    quantity INTEGER NOT NULL CHECK (quantity >= 0),
    PRIMARY KEY (warehouse_id, book_id),
    CONSTRAINT FK_Book_x_WarehouseWarehouse FOREIGN KEY (warehouse_id)
REFERENCES bookstore.Warehouse(warehouse_id),
    CONSTRAINT FK_Book_x_WarehouseBook FOREIGN KEY (book_id) REFERENCES
bookstore.Book(book_id)
);

-- Table Warehouse_X_Delivery_service (склад-служба доставки)
CREATE TABLE bookstore.Warehouse_x_Delivery_service (
    warehouse_id INTEGER NOT NULL,
    delivery_service_id INTEGER NOT NULL,
    PRIMARY KEY (warehouse_id, delivery_service_id),
    CONSTRAINT FK_Warehouse_x_Delivery_serviceWarehouse FOREIGN KEY
(warehouse_id) REFERENCES bookstore.Warehouse(warehouse_id),
    CONSTRAINT FK_Warehouse_x_Delivery_serviceDelivery_service FOREIGN KEY
(delivery_service_id) REFERENCES
bookstore.Delivery_service(delivery_service_id)
);
```

```sql
-- Table Book_x_Sale (книга-продажа)
CREATE TABLE bookstore.Book_x_Sale (
    book_id INTEGER NOT NULL,
    sale_id INTEGER NOT NULL,
    sale_price NUMERIC(15, 2) NOT NULL CHECK (sale_price >= 0),
    quantity INTEGER NOT NULL CHECK (quantity >= 0),
    PRIMARY KEY (book_id, sale_id),
    CONSTRAINT FK_Book_x_SaleBook FOREIGN KEY (book_id) REFERENCES
bookstore.Book(book_id),
    CONSTRAINT FK_Book_x_SaleSale FOREIGN KEY (sale_id) REFERENCES
bookstore.Sale(sale_id)
);

-- Table Warehouse_x_Publishing (склад-издательство)
CREATE TABLE bookstore.Warehouse_x_Publishing (
    warehouse_id INTEGER NOT NULL,
    publishing_id INTEGER NOT NULL,
    PRIMARY KEY (warehouse_id, publishing_id),
    CONSTRAINT FK_Warehouse_x_PublishingWarehouse FOREIGN KEY
(warehouse_id) REFERENCES bookstore.Warehouse(warehouse_id),
    CONSTRAINT FK_Warehouse_x_PublishingPublishing FOREIGN KEY
(publishing_id) REFERENCES bookstore.Publishing(publishing_id)
);

-- Table Book_x_Genre (книга-жанр)
CREATE TABLE bookstore.Book_x_Genre (
    genre_id INTEGER NOT NULL,
    book_id INTEGER NOT NULL,
    PRIMARY KEY (genre_id, book_id),
    CONSTRAINT FK_Book_x_GenreBook FOREIGN KEY (book_id) REFERENCES
bookstore.Book(book_id),
    CONSTRAINT FK_Book_x_GenreGenre FOREIGN KEY (genre_id) REFERENCES
bookstore.Genre(genre_id)
);
```

# Задание 4

```
INSERT INTO bookstore.Publishing(name, address, city) VALUES
    ('Лабиринт', '2-й Рощинский проезд, 8с4', 'Москва'),
    ('Эксмо', 'улица Зорге, дом 1', 'Москва'),
    ('Росмэн', 'Ленинградский проспект, д. 36', 'Москва'),
    ('Просвещение', 'Краснопролетарская, д. 16', 'Москва'),
    ('Bubble Comics', 'ул Бутырский Вал, д 68/70', 'Москва'),
    ('Алтапресс', 'Короленко, 107', 'Барнаул'),
    ('Ленинградское издательство', 'Менделеевская ул., 9',
'Сантк-Петербург'),
    ('Речь', 'лн. 11-я В.О., д. 26', 'Сантк-Петербург'),
    ('Татарское книжное издательство', 'ул. Баумана, 51', 'Казань'),
    ('Калининградская книга', 'ул. Карла Маркса, 18', 'Калининград');


INSERT INTO bookstore.Warehouse(city, address, phone_number, capacity,
current_stock) VALUES
    ('Москва', 'улица Зорге, дом -100', '+7-111-111-11-11', 500, 66),
    ('Москва', 'Короленко, -50', '+7-222-222-22-22', 100, 96),
    ('Москва', 'улица Зорге, дом -200', '+7-333-333-33-33', 2000, 212),
    ('Москва', 'ул. Баумана, 0', '+7-444-444-44-44', 200, 110),
    ('Казань', 'Петербургская, -1', '+7-555-555-55-55', 3456, 187),
    ('Калининград', 'ул. Карла Маркса, -18', '+7-666-666-66-66', 1386,
39),
    ('Барнаул', 'Короленко, 1005', '+7-777-777-77-77', 1005, 500),
    ('Санкт-Петербург', 'ул. Замшина, д. 10000', '+7-888-888-88-88', 533,
239);


INSERT INTO bookstore.Delivery_service (name) VALUES
    ('cdek'),
    ('Dpd'),
    ('ЯндексДоставка'),
    ('spb.dostavista'),
    ('Boxberry');
```

```
INSERT INTO bookstore.Buyer(name, email, phone_number, valid_from_DTTM,
valid_to_DTTM) VALUES
    ('Диана', 'mediana@work.ru', '+7-999-999-99-99', '2020-12-01',
'2022-12-05'),
    ('Диана', 'mediana@work.ru', '+7-000-000-00-00', '2022-12-05',
'2024-12-05'),
    ('Диана', 'mediana105@work.ru', '+7-000-000-00-00', '2024-12-05',
NULL),
    ('Екатерина', 'isaeva@work.ru', '+7-123-999-99-99', '2024-12-01',
NULL),
    ('Алина', 'mukha@work.ru', '+7-123-000-00-00', '2020-06-05',
'2022-11-11'),
    ('Алина', 'mukha@work.ru', '+7-111-000-00-00', '2022-11-11', NULL),
    ('Petya Petrov', 'petya@yandex.ru', '+7-111-111-00-00', '2020-01-01',
'2022-01-11'),
    ('Vasya Vasiliev', 'vasya@yandex.ru', '+7-111-111-11-00',
'2022-01-01', NULL),
    ('Ivan Ivanov', 'vanya@yandex.ru', '+7-222-222-22-00', '2020-06-01',
NULL),
    ('Igor', 'igor@work.com', '+7-111-111-11-22', '2023-05-01', NULL);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (1, 'Теория всего', 'Стивен Хокинг', 1500),
    (1, 'Алиса в Зазеркалье', 'Льюис Кэрролл', 670),
    (1, 'Записки юного врача', 'М.А.Булгаков', 500);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (2, 'Цветы для Элджернона', 'Дэниел Киз', 450),
    (2, 'Преступление и наказание', 'Ф.М.Достоевский', 430),
    (2, 'Маленький принц', 'А.Д.Сент-Экзюпери', 500),
    (2, 'Алиса в Зазеркалье', 'Льюис Кэрролл', 900);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (3, 'Гарри Поттер и философский камень', 'Дж. К. Роулинг', 1000),
    (3, 'Гарри Поттер и Тайная комната', 'Дж. К. Роулинг', 1200),
    (3, 'Гарри Поттер и узник Азкабана', 'Дж. К. Роулинг', 1150),
    (3, 'Гарри Поттер и Кубок огня', 'Дж. К. Роулинг', 890),
    (3, 'Гарри Поттер и Орден Феникса', 'Дж. К. Роулинг', 1100),
    (3, 'Гарри Поттер и Принц-полукровка', 'Дж. К. Роулинг', 990),
    (3, 'Гарри Поттер и Дары Смерти', 'Дж. К. Роулинг', 900);
```

```sql
INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (4, 'Всеобщая история 6 класс', 'Е.А.Крючкова', 300),
    (4, 'Немецкий язык 9 класс', 'М.М.Аверин', 400),
    (4, 'Математика 2 класс', 'В.Н.Рудницкая', 350);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (5, 'Комикс Чумной доктор. Том 1. Капкан', 'Наталия Воронцова', 600),
    (5, 'Ловец бабочек: книга комиксов', 'Анастасия Ким', 750),
    (5, 'Инок. Наследие', 'Роман Котков', 500),
    (5, 'Найди Дракона', 'Роман Котков', 500);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (6, 'Поющая радуга', 'Ирина Цхай', 300);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (7, 'Заря противоборства', 'Андрей Один', 500),
    (7, 'Планета луунов', 'Живой А.Я.', 400);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (8, 'Девочка-свеча', 'Софья Прокофьева', 700);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (9, 'Су анасы', 'Габдулла Тукай', 400);

INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (10, 'Мальчик с янтарного берега', 'Мартин Бергау', 300);
```

```sql
INSERT INTO bookstore.Genre(name) VALUES
    ('Детектив'),
    ('Фантастика'),
    ('Фэнтези'),
    ('Реализм'),
    ('Триллер'),
    ('Ужасы'),
    ('Классика'),
    ('Драма'),
    ('Комедия'),
    ('Поэзия'),
    ('Учебная литература'),
    ('Сказка'),
    ('Трагедия'),
    ('Роман'),
    ('Рассказы'),
    ('Комиксы');

INSERT INTO bookstore.Sale(buyer_id, delivery_service_id, warehouse_id,
address, DTTM) VALUES
    (1, 1, 1, 'г. Москва, ул. Карла Маркса, 1', '2021-12-12'),
    (2, 3, 2, 'г. Набережные Челны, 18/01',  '2024-11-01'),
    (2, 3, 2, 'г. Казань, ул. Петербургская, 1','2024-09-12'),
    (4, 1, 7, 'г. Москва, улица Зорге, 1','2024-12-01'),
    (5, 4, 8, 'г. Санкт-Петербург, ул. Кантемировская', '2021-11-11'),
    (1, 1, 1, 'г. Москва, ул. Карла Маркса, 2', '2022-12-12'),
    (1, 2, 1, 'г. Москва, ул. Карла Маркса, 2', '2022-12-11'),
    (2, 2, 1, 'г. Набережные Челны, 18/01', '2024-11-02');

INSERT INTO bookstore.Book_x_Warehouse(warehouse_id, book_id, quantity)
VALUES
    (1, 1, 10), (1, 5, 5), (1, 7, 4), (1, 10, 5), (1, 11, 8), (1, 16, 20),
(1, 18, 4), (1, 19, 10),
    (2, 6, 17), (2, 7, 50), (2, 15, 4), (2, 16, 6), (2, 17, 19),
    (3, 2, 101), (3, 5, 20), (3, 18, 11), (3, 20, 45), (3, 21, 5), (3, 3,
30),
    (4, 7, 5), (4, 8, 4), (4, 9, 1), (4, 10, 100),
    (5, 26, 30), (5, 12, 4), (5, 13, 18), (5, 1, 35), (5, 2, 100),
    (6, 27, 4), (6, 15, 5), (6, 16, 6), (6, 17, 7), (6, 13, 8), (6, 14,
9),
    (7, 22, 100), (7, 4, 200), (7, 5, 100), (7, 6, 100),
    (8, 1, 100), (8, 5, 48), (8, 9, 52), (8, 15, 39);
```

```
INSERT INTO bookstore.Warehouse_x_Delivery_service(warehouse_id,
delivery_service_id) VALUES
    (1, 1), (1, 2), (2, 1), (2, 3), (2, 5), (3, 1), (4, 1), (4, 5), (5,
1),
    (5, 2), (5, 3), (6, 1), (6, 3), (7, 1), (7, 3), (8, 1), (8, 4), (8,
5);

INSERT INTO bookstore.Book_x_Sale(book_id, sale_id, sale_price, quantity)
VALUES
    (1, 1, 1500, 2), (7 , 2, 800, 1), (6 , 2, 500, 1), (8, 3, 1000, 2),
(22, 4, 400, 1), (15, 5, 300, 1),
    (9, 5, 200, 1), (1, 6, 800, 20), (1, 7, 300, 5), (7, 8, 800, 1);


INSERT INTO bookstore.Warehouse_x_Publishing(warehouse_id, publishing_id)
VALUES
    (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 2), (2, 4), (3, 1), (3,
2), (3, 5), (4, 2), (4, 3),
    (5, 9), (5, 1), (5, 3), (6, 10), (6, 4), (6, 3), (7, 6), (7, 2), (8,
1), (8, 2), (8, 3), (8, 4);

INSERT INTO bookstore.Book_x_Genre(book_id, genre_id) VALUES
    (1, 11), (2, 3), (3, 15), (3, 7), (4, 14), (4, 2), (5, 7), (5, 14),
(5, 8), (6, 3), (6, 12),
    (7, 3), (8, 3), (8, 14), (9, 3), (9, 14), (10, 3), (10, 14), (11, 3),
(11, 14), (12, 3), (12, 14),
    (13, 3), (13, 14), (14, 3), (14, 14), (15, 11), (16, 11), (17, 11),
(18, 16), (18, 4), (18, 8), (19, 3),
    (19, 16), (20, 3), (20, 16), (21,3), (21,16), (22, 12), (23, 14), (24,
2), (25, 15), (26, 12), (27, 8), (27, 4);
```

**Задание 5**

```sql
-- CRUD-запросы (INSERT, SELECT, UPDATE, DELETE) к двум таблицам БД:
bookstore.Buyer и  bookstore.Book

-- Добавить нового пользователя в таблицу bookstore.Buyer
 INSERT INTO bookstore.Buyer(name, email, phone_number, valid_from_DTTM,
valid_to_DTTM) VALUES
    ('Tanya', 'tanya@work.ru', '+7-012-345-67-89', CURRENT_DATE, NULL);

-- Добавить новую книгу в таблицу bookstore.Book
INSERT INTO bookstore.Book(publishing_id, name, author, price) VALUES
    (1, 'Приключения Алисы в Стране Чудес', 'Льюис Кэрролл', 3000);


-- Вывести все записи с покупателями, действительные на данный момент
SELECT * FROM bookstore.Buyer WHERE valid_to_DTTM IS NULL;

-- Вывести название, автора и цену для всех книг ценой меньше 1000,
отсортировать в порядке возрастания цены
SELECT name, author, price FROM bookstore.Book WHERE price < 1000 ORDER BY
price;

-- Вывести все книги Льюиса Кэрролла, в названии которых есть имя "Алиса"
в любом склонении
SELECT * FROM bookstore.Book WHERE author = 'Льюис Кэрролл' AND name LIKE
'%Алис%';

-- Посчитать количество записей, актуальных на начало 2022 года, в таблице
покупателей
SELECT COUNT(*)  FROM bookstore.Buyer WHERE valid_from_DTTM < '2022-01-01'
AND '2022-01-01' <= valid_to_DTTM;

-- Посчитать количество разных книг в каждом издательстве
-- Вывести id издательства и количество книг, отсортировать по убыванию
количества
SELECT publishing_id, COUNT(*) FROM bookstore.Book GROUP BY(publishing_id)
ORDER BY COUNT(*) DESC;

-- Поднять на 100 цену книги "Преступление и наказание", изданной
издательством с id = 2
UPDATE bookstore.Book SET price = price + 100 WHERE name = 'Преступление и
наказание' AND publishing_id = 2;
```

```sql
-- Снизить на 10% цену всех книг пятого издательства
UPDATE bookstore.Book SET price = price * 0.9 WHERE publishing_id = 5;

-- Поменять автора книг с "Ф.М.Достоевский" на "Фёдор Михайлович Достоевский"
UPDATE bookstore.Book SET author = 'Фёдор Михайлович Достоевский' WHERE author = 'Ф.М.Достоевский';
```

# Задание 6

```sql
-- 1. В результате выполнения запроса будут получены самые дорогие книги для каждого склада в Москве или Санкт-Петербурге.
-- Если таких книг несколько, будут выведены все из них.
-- Формат вывода: id, название и стоимость  книги, id склада, город, в котором расположен склад

SELECT w_id, city, b_id, name, CASE WHEN price IS NULL THEN 0 ELSE price END AS price
FROM (
    SELECT  b.book_id AS b_id,
            b.name AS name,
            b.price AS price,
            w.warehouse_id as w_id,
            w.city as city,
            rank() OVER (PARTITION BY w.warehouse_id ORDER BY b.price DESC) as place

    FROM bookstore.Warehouse AS w
    LEFT JOIN bookstore.Book_x_Warehouse AS bw ON bw.warehouse_id = w.warehouse_id
    LEFT JOIN bookstore.Book AS b ON b.book_id = bw.book_id
    WHERE w.city = 'Москва' OR w.city = 'Санкт-Петербург'
) AS sorted_books
WHERE sorted_books.place = 1;
```

```sql
-- 2. В результате выполнения запроса будет получена общая стоимость всех заказов, сделанных в каждом году
-- (в котором был сделан хотя бы 1 заказ), и сумма, на которую заказы отличаются от предыдущего года
-- Формат вывода: год, суммарная стоимость заказов за этот год, разность с суммой за предыдущий год.

WITH sum_by_year AS (
    SELECT  year, SUM(sale_price * quantity) AS total_price
    FROM (
        SELECT  EXTRACT(YEAR FROM s.DTTM) AS year,
                bs.sale_price AS sale_price,
                bs.quantity AS  quantity
        FROM bookstore.Sale AS s INNER JOIN bookstore.Book_x_Sale AS bs ON bs.sale_id = s.sale_id
    ) AS year_info
    GROUP BY year
)

SELECT year,
        total_price,
        CASE WHEN prev_year = year - 1
            THEN total_price - prev_price
            ELSE total_price
        END as diff
FROM (
    SELECT  year,
            total_price,
            lag(year) OVER (ORDER BY year ASC) AS prev_year,
            lag(total_price) OVER (ORDER BY year ASC) AS prev_price
    FROM sum_by_year
) AS perv_year_price;
```

```sql
-- 3. В результате выполнения запроса будут получены склады, доставка с
которых производилась преимущественно
-- компанией "cdek"
-- Формат вывода: id склада, количество доставок с этого склада cdek-ом,
процент, который составляют
-- эти доставки от всех доставок с этого склада
WITH delivery_in_sale AS (
    SELECT  w.warehouse_id AS w_id,
            s.delivery_service_id AS d_id,
            COUNT(*) AS delivery_cnt,
            d.name as delivery_name
    FROM bookstore.Warehouse AS w
    INNER JOIN bookstore.sale AS s ON s.warehouse_id = w.warehouse_id
    INNER JOIN bookstore.delivery_service AS d ON d.delivery_service_id =
s.delivery_service_id
    GROUP BY w.warehouse_id, s.delivery_service_id, d.name
)

SELECT w_id, delivery_cnt, ROUND(delivery_cnt * 100 /
all_deliveries_cnt::NUMERIC, 2) AS percent
FROM (
    SELECT  w_id,
            d_id,
            delivery_name,
            delivery_cnt,
            SUM(delivery_cnt) OVER (PARTITION BY w_id) as
all_deliveries_cnt,
            rank() OVER(PARTITION BY w_id ORDER BY delivery_cnt DESC) as
place
    FROM delivery_in_sale
) AS deliveries_cnt
WHERE delivery_name = 'cdek' AND place = 1;
```

```sql
-- 4. В результате выполнения запроса будут получены все покупатели,
которые покупали учебную литературу не реже других жанров
-- Формат вывода: id покупателя, имя покупателя, количество купленных им
учебных книг и их список через запятую
WITH genres_cnt AS (
    SELECT  buyer.buyer_id AS buyer_id,
            buyer.name AS buyer_name,
            g.genre_id AS genre_id,
            g.name AS genre_name,
            SUM (bs.quantity) AS genre_sales_cnt

    FROM bookstore.Buyer as buyer
    INNER JOIN bookstore.Sale AS s ON s.buyer_id = buyer.buyer_id
    INNER JOIN bookstore.Book_x_Sale AS bs ON bs.sale_id = s.sale_id
    INNER JOIN bookstore.Book AS b ON bs.book_id = b.book_id
    INNER JOIN bookstore.Book_x_Genre AS bg ON bg.book_id = b.book_id
    INNER JOIN bookstore.Genre AS g ON bg.genre_id = g.genre_id

    GROUP BY buyer.buyer_id, buyer.name, g.genre_id, g.name
)

SELECT  buyer_id, buyer_name, genre_sales_cnt,
        string_agg(b.name, ', ') AS books_list
FROM (
    SELECT  buyer_id, buyer_name, genre_id, genre_name, genre_sales_cnt,
            rank() OVER (PARTITION BY buyer_id ORDER BY genre_sales_cnt
DESC) AS place
    FROM genres_cnt
) AS genres_places

LEFT JOIN bookstore.Book_x_Genre AS bg ON bg.genre_id =
genres_places.genre_id
LEFT JOIN bookstore.Book AS b ON bg.book_id = b.book_id

WHERE genre_name = 'Учебная литература' AND place = 1
GROUP BY buyer_id, buyer_name, genre_sales_cnt;
```

```sql
-- 5. В результате выполнения запроса будет получен список всех актуальных на данный момент покупателей, сумма стоимостей
-- совершенных ими покупок и сумма, на которую стоимость их покупок отличается от максимальной. Список отсортирован по стоимости трат.
-- Формат вывода: id покупателя, имя покупателя, сумма его трат, разница между максимальной суммой и суммой его покупок
WITH sales_sum AS (
    SELECT  buyer.buyer_id AS buyer_id,
            buyer.name AS buyer_name,
            SUM (bs.sale_price * bs.quantity) AS buyer_price
    FROM bookstore.Buyer AS buyer
    LEFT JOIN bookstore.Sale AS s ON buyer.buyer_id = s.buyer_id
    LEFT JOIN bookstore.Book_x_Sale AS bs ON s.sale_id = bs.sale_id

    WHERE buyer.valid_to_dttm IS NULL
    GROUP BY buyer.buyer_id, buyer.name
)

SELECT buyer_id,
        buyer_name,
        buyer_purchase_price,
        CASE WHEN max_purchase_price IS NULL THEN 0 ELSE
max_purchase_price - buyer_purchase_price END AS diff
FROM (
    SELECT  buyer_id,
            buyer_name,
            CASE WHEN buyer_price IS NULL THEN 0 ELSE buyer_price END AS
buyer_purchase_price,
            first_value(buyer_price) OVER (ORDER BY buyer_price DESC NULLS
LAST) AS max_purchase_price
    FROM sales_sum
) AS sum_and_first_value
ORDER BY buyer_purchase_price;
```

# Задание 7

```sql
CREATE SCHEMA views;

-- Publishing View
CREATE VIEW views.Publishing_view AS
SELECT
    publishing_id,
    name,
    address,
    city
FROM bookstore.Publishing;

-- Warehouse View
CREATE VIEW views.Warehouse_view AS
SELECT
    warehouse_id,
    city,
    address,
    phone_number,
    capacity,
    current_stock
FROM bookstore.Warehouse;

-- Delivery Service View
CREATE VIEW views.Delivery_service_view AS
SELECT
    delivery_service_id,
    name
FROM bookstore.Delivery_service;
```

```sql
-- Buyer View
CREATE VIEW views.Buyer_view AS
SELECT
    buyer_id,
    CASE
        WHEN LENGTH(name) <= 2 THEN REPEAT('*', LENGTH(name))
        ELSE CONCAT(SUBSTRING(name, 1, 3), REPEAT('*', LENGTH(name) - 3))
    END AS name,
    CONCAT(REPEAT('*', POSITION('@' IN email) - 1), SUBSTRING(email,
POSITION('@' IN email))) AS email,
    CONCAT(SUBSTRING(phone_number, 1, 4), '********') AS phone_number,
    valid_from_DTTM,
    valid_to_DTTM
FROM bookstore.Buyer;

-- Genre View
CREATE VIEW views.Genre_view AS
SELECT
    genre_id,
    name
FROM bookstore.Genre;

-- Book View
CREATE VIEW views.Book_view AS
SELECT
    book_id,
    name,
    author,
    price
FROM bookstore.Book

-- Sale View
CREATE VIEW views.Sale_view AS
SELECT
    sale_id,
    buyer_id,
    delivery_service_id,
    warehouse_id,
    address,
    DTTM
FROM bookstore.Sale;
```

```sql
-- Book x Warehouse View
CREATE VIEW views.Book_x_Warehouse_view AS
SELECT
    bw.book_id,
    b.name AS book_name,
    bw.warehouse_id,
    w.city AS warehouse_city,
    w.address AS warehouse_address,
    bw.quantity
FROM bookstore.Book_x_Warehouse bw
LEFT JOIN bookstore.Book b ON bw.book_id = b.book_id
LEFT JOIN bookstore.Warehouse w ON bw.warehouse_id = w.warehouse_id;

-- Warehouse x Delivery Service View
CREATE VIEW views.Warehouse_X_Delivery_service_view AS
SELECT
    wds.warehouse_id,
    w.city AS warehouse_city,
    w.address AS warehouse_address,
    wds.delivery_service_id,
    ds.name AS delivery_service_name
FROM bookstore.Warehouse_x_Delivery_service wds
LEFT JOIN bookstore.Warehouse w ON wds.warehouse_id = w.warehouse_id
LEFT JOIN bookstore.Delivery_service ds ON wds.delivery_service_id =
ds.delivery_service_id;

-- Book x Sale View
CREATE VIEW views.Book_x_Sale_view AS
SELECT
    bs.book_id,
    b.name AS book_name,
    bs.sale_id,
    s.DTTM AS sale_date,
    bs.sale_price,
    bs.quantity
FROM bookstore.Book_x_Sale bs
LEFT JOIN bookstore.Book b ON bs.book_id = b.book_id
LEFT JOIN bookstore.Sale s ON bs.sale_id = s.sale_id;
```

```sql
-- Warehouse x Publishing View
CREATE VIEW views.Warehouse_x_Publishing_view AS
SELECT
    wp.warehouse_id,
    w.city AS warehouse_city,
    w.address AS warehouse_address,
    wp.publishing_id,
    p.name AS publishing_name
FROM bookstore.Warehouse_x_Publishing wp
LEFT JOIN bookstore.Warehouse w ON wp.warehouse_id = w.warehouse_id
LEFT JOIN bookstore.Publishing p ON wp.publishing_id = p.publishing_id;

-- Book x Genre View
CREATE VIEW views.Book_x_Genre_view AS
SELECT
    bg.book_id,
    b.name AS book_name,
    bg.genre_id,
    g.name AS genre_name
FROM bookstore.Book_x_Genre bg
LEFT JOIN bookstore.Book b ON bg.book_id = b.book_id
LEFT JOIN bookstore.Genre g ON bg.genre_id = g.genre_id;
```

## Задание 8

```sql
-- Представление, которое показывает общие продажи и количество продаж для каждой книги,
-- а также разницу в количестве между текущим и предыдущим значениями по количеству проданных книг.

DROP VIEW IF EXISTS views.books_total_sales;
CREATE VIEW views.books_total_sales AS
WITH sales_each_book AS (
    SELECT
        b.name AS book_name,
        b.author,
        bs.sale_price AS book_price,
        COALESCE(SUM(bs.quantity * bs.sale_price), 0.00) AS total_sales,
        COALESCE(SUM(bs.quantity), 0) AS total_quantity
    FROM
        bookstore.Book AS b
    LEFT JOIN
        bookstore.Book_x_Sale AS bs ON b.book_id = bs.book_id
    GROUP BY
        b.book_id, b.name, b.author, bs.sale_price
)
SELECT
    seb.book_name,
    seb.author,
    seb.book_price,
    seb.total_sales,
    seb.total_quantity,
        COALESCE(LAG(seb.total_quantity) OVER (ORDER BY seb.total_quantity DESC), 0) AS prev_total_quantity,
    COALESCE(seb.total_quantity, 0) - COALESCE(LAG(seb.total_quantity) OVER (ORDER BY seb.total_quantity DESC), 0) AS difference
FROM sales_each_book AS seb
ORDER BY seb.total_quantity DESC;
```

```sql
-- Представление, которое показывает для каждой книги дату с наибольшим количеством продаж. В случае, если несколько дней имеют одинаковое количество продаж, все такие дни будут выведены.

DROP VIEW IF EXISTS views.top_days_per_book;
CREATE VIEW views.top_days_per_book AS
WITH sales_details AS (
    SELECT
        b.book_id,
        b.name AS book_name,
        s.DTTM::DATE AS sale_date,
        COUNT(s.sale_id) AS total_count
    FROM
        bookstore.Book AS b
    LEFT JOIN
        bookstore.Book_x_Sale AS bs ON b.book_id = bs.book_id
    LEFT JOIN
        bookstore.Sale AS s ON bs.sale_id = s.sale_id
    GROUP BY
        b.book_id, b.name, s.DTTM::DATE
),
most_frequent_day AS (
    SELECT
        book_id,
        sale_date,
        RANK() OVER (PARTITION BY book_id ORDER BY total_count DESC) AS day_rank
    FROM
        sales_details
)
SELECT
    sd.book_id,
    sd.book_name,
    mfd.sale_date
FROM
    sales_details AS sd
LEFT JOIN
    most_frequent_day AS mfd ON sd.book_id = mfd.book_id AND sd.sale_date = mfd.sale_date
WHERE
    mfd.day_rank = 1
ORDER BY
    sd.book_name, mfd.day_rank;
```

```sql
-- Представление, содержащее для каждого покупателя список складов, с которых он чаще всего совершал покупки.
-- В случае, если несколько складов являются наиболее частыми, все такие склады будут выведены.
DROP VIEW IF EXISTS views.most_frequent_warehouse_for_buyers;
CREATE VIEW views.most_frequent_warehouse_for_buyers AS
WITH buyer_x_Warehouse_statistics AS (
    SELECT
        s.buyer_id,
        s.warehouse_id,
        COUNT(s.sale_id) AS sales_count
    FROM
        bookstore.Sale AS s
    GROUP BY
        s.buyer_id, s.warehouse_id
),
top_warehouses AS (
    SELECT
        bws.buyer_id,
        bws.warehouse_id,
        bws.sales_count,
        RANK() OVER (PARTITION BY bws.buyer_id ORDER BY bws.sales_count DESC) AS rnk
    FROM
        buyer_x_Warehouse_statistics AS bws
)
SELECT
    b.name AS buyer_name,
    tw.warehouse_id,
    tw.sales_count
FROM
    top_warehouses AS tw
INNER JOIN
    bookstore.Buyer AS b ON tw.buyer_id = b.buyer_id
WHERE
    tw.rnk = 1
ORDER BY
    b.name, tw.sales_count DESC;
```

## Задание 9

```
-- Триггер над Book_x_Sale, уменьшающий текущую заполненность склада, если добавляется новый заказ
CREATE OR REPLACE FUNCTION bookstore.update_current_stock()
RETURNS TRIGGER AS $$
BEGIN

       UPDATE bookstore.Warehouse
       SET current_stock = current_stock - NEW.quantity
       WHERE warehouse_id =
       (
           SELECT warehouse_id
           FROM bookstore.Sale
           WHERE sale_id = NEW.sale_id
       );

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update_current_stock_trigger
AFTER INSERT ON bookstore.Book_x_Sale
FOR EACH ROW
EXECUTE FUNCTION bookstore.update_current_stock();


-- Триггер над Buyer, запрещающий стандартное обновление таблицы Buyer
CREATE OR REPLACE FUNCTION bookstore.update_buyer()
RETURNS TRIGGER AS $$
BEGIN
   RAISE EXCEPTION 'Update prohibited';
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update_buyer_trigger
BEFORE UPDATE ON bookstore.Buyer
FOR EACH ROW
EXECUTE FUNCTION bookstore.update_buyer();
```

## Задание 10

```
-- Процедура, добавляющая новый заказ
-- call bookstore.add_sale(4, 1, 1, 'г.Санкт-Петербург, ул. Катемировская, д. 3а', NOW()::TIMESTAMP, '[{"book_id": 1, "quantity": 2, "sale_price": 10.00}, {"book_id": 5, "quantity": 1, "sale_price": 10.00}]'::JSONB);
CREATE OR REPLACE PROCEDURE bookstore.add_sale(
    buyer_id INTEGER,
    delivery_service_id INTEGER,
    warehouse_id_from INTEGER,
    address VARCHAR,
    current_dttm TIMESTAMP,
    books JSONB
)
LANGUAGE plpgsql
AS $$
DECLARE
    new_sale_id INTEGER;
    total_quantity INTEGER = 0;
    book JSONB;
BEGIN
    -- Проверка количества книг на складе
    FOR book IN SELECT * FROM jsonb_array_elements(books) LOOP
        total_quantity := total_quantity + (book->>'quantity')::INTEGER;
    END LOOP;

    IF total_quantity > (SELECT current_stock from Warehouse WHERE warehouse_id=warehouse_id_from)
        THEN RAISE EXCEPTION 'Not enough stock in warehouse';
    END IF;
    -- Добавление новой продажи в Sale
    INSERT INTO bookstore.Sale (buyer_id, delivery_service_id, warehouse_id, address, DTTM)
    VALUES (buyer_id, delivery_service_id, warehouse_id_from, address, current_dttm)
    RETURNING sale_id INTO new_sale_id;
```

```
    -- Обновление количества книг из заказа на складе
    FOR book IN SELECT * FROM jsonb_array_elements(books) LOOP
        DECLARE
            book_sale_id INTEGER := (book->>'book_id')::INTEGER;
            book_quantity INTEGER := (book->>'quantity')::INTEGER;
            book_sale_price NUMERIC := (book->>'sale_price')::NUMERIC;
        BEGIN
            -- Добавление в Book_x_Sale

            INSERT INTO bookstore.Book_x_Sale (book_id, sale_id,
sale_price, quantity)
            VALUES (book_sale_id, new_sale_id, book_sale_price,
book_quantity);

            -- Обновление количества определенной книги в Warehouse
            UPDATE bookstore.Book_x_Warehouse
            SET quantity = quantity - book_quantity
            WHERE bookstore.Book_x_Warehouse.warehouse_id =
warehouse_id_from AND book_id = book_sale_id;
        END;
    END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            RAISE NOTICE 'Not enough books in warehouse';
            ROLLBACK;
END;
$$;
```

```sql
-- Процедура, добавляющая новую книгу
-- call bookstore.add_book(1, 'Доктор Живаго', 'Пастернак Б.Л.', 300.99,
'[{"warehouse_id": 1, "quantity": 4}, {"warehouse_id": 5, "quantity":
500}]'::JSONB, '{1, 5, 14}');
CREATE OR REPLACE PROCEDURE bookstore.add_book(
    publishing_id INTEGER,
    name VARCHAR,
    author VARCHAR,
    price NUMERIC,
    warehouses JSONB,
    genres INTEGER ARRAY
)
LANGUAGE plpgsql
AS $$
DECLARE
    new_book_id INTEGER;
    warehouse JSONB;
    genre INTEGER;
BEGIN
    -- Добавление в Book
    INSERT INTO bookstore.Book(publishing_id, name, author, price)
    VALUES (publishing_id, name, author, price)
    RETURNING book_id INTO new_book_id;

    FOR warehouse IN SELECT * FROM jsonb_array_elements(warehouses) LOOP
        DECLARE
            warehouse_id INTEGER := (warehouse->>'warehouse_id')::INTEGER;
            book_quantity INTEGER := (warehouse->>'quantity')::INTEGER;
        BEGIN
            -- Добавление в Book_x_Warehouse
            INSERT INTO bookstore.Book_x_Warehouse (warehouse_id, book_id,
quantity)
            VALUES (warehouse_id, new_book_id, book_quantity);

            -- Обновление текущей заполненности склада
            UPDATE bookstore.Warehouse
            SET current_stock = current_stock + book_quantity
            WHERE bookstore.Warehouse.warehouse_id = warehouse_id;
        END;
    END LOOP;
```

```
    FOREACH genre IN ARRAY genres LOOP
        -- Добавление в Book_x_genre
        INSERT INTO bookstore.Book_x_genre (book_id, genre_id)
        VALUES (new_book_id, genre);
    END LOOP;

    EXCEPTION
        WHEN OTHERS THEN
            RAISE NOTICE 'Not enough capacity in warehouse';
            ROLLBACK;
END;
$$;
```

# Задание 11

```
from peewee import *
import config

bookStoreDB = PostgresqlDatabase(config.DATABASE_NAME,
user=config.DATABASE_USER, password=config.DATABASE_PASSWORD,
host=config.DATABASE_HOST, port=config.DATABASE_PORT)

class BaseModel(Model):
    class Meta:
        database = bookStoreDB
        schema = 'bookstore'


class Book(BaseModel):
    book_id = AutoField()
    publishing_id = IntegerField()
    name = CharField()
    author = CharField()
    price = DecimalField()


class Warehouse(BaseModel):
    warehouse_id = AutoField()
    city = CharField()
    address = CharField()
    phone_number = CharField()
    capacity = IntegerField()
    current_stock = IntegerField()


class Genre(BaseModel):
    genre_id = AutoField()
    name = CharField()


class Publishing(BaseModel):
    publishing_id = AutoField()
    name = CharField()
    address = CharField()
    city = CharField()
```

```python
class Book_x_Sale(BaseModel):
    book_id = IntegerField()
    sale_id = IntegerField()
    sale_price = DecimalField()
    quantity = IntegerField()


# Получение всех номеров телефонов для складов в конкретном городе
def get_phone_numbers(city: str) -> list:
    numbers: list = []
    for warehouse in Warehouse.select().where(Warehouse.city == city):
        numbers.append([warehouse.warehouse_id, warehouse.city,
warehouse.phone_number, warehouse.address])
    return numbers

# print(get_phone_numbers('Москва'))
# print(get_phone_numbers('Екатеринбург'))


# Добавление нового жанра
def insert_genre(genre_name: str):
    Genre.create(name=genre_name)

# insert_genre('Подростковая проза')


# Обновление адреса издательства
def update_address(new_address: str, publishing_name: str):

Publishing.update(address=new_address).where(Publishing.name==publishing_n
ame).execute()

# update_address('ул. Карла Маркса, 19', 'Калининградская книга')
```

```python
# Топ книг по количеству продаж
def get_top_books():
    query = (Book
             .select(Book.name, fn.COALESCE(fn.SUM(Book_x_Sale.quantity),
0).alias('total_quantity'),

fn.DENSE_RANK().over(order_by=[fn.COALESCE(fn.SUM(Book_x_Sale.quantity),
0).desc()]).alias('place'))
             .join(Book_x_Sale, JOIN.LEFT_OUTER,
on=(Book.book_id==Book_x_Sale.book_id))
             .group_by(Book.name)
             .order_by(fn.COALESCE(fn.SUM(Book_x_Sale.quantity), 0).desc()))
    for book in query:
        print(book.name, book.total_quantity, book.place)

# get_top_books()
```