МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Базы данных»
Тема: Реализация базы данных в СУБД PostgreSQL

Студент гр. 3341 _____ Шаповаленко Е.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2025

Цель работы.

Целью данной лабораторной работы является практическое освоение навыков создания и администрирования баз данных в среде СУБД PostgreSQL. Для достижения цели необходимо написать запросы для создания таблиц из предыдущей лабораторной работы и выполнения ряда действий, указанных в задании к ней.

Задание.

Необходимо развернуть PostgreSQL локально:

- Написать запросы для создания таблиц из предыдущей лабораторный работы
- Заполнить тестовыми данными: 5-10 строк на каждую таблицу, обязательно наличие связи между ними, данные приближены к реальности.
- Написать запросы к БД, отвечающие на вопросы из предыдущей лабораторной работы
- Исходный код выложить на www.db-fiddle.com для проверки работоспособности
- Исходный код в виде .sql файла запушить в виде PR в репо
- В отчете описать:
- Цель
- Текст задания в соответствии с вариантом
- Скриншоты работы с СУБД PostgreSQL (psql / DBeaver / Datagrip, ...)
- Скриншоты на каждый запрос (или группу запросов) на изменение/таблицы с выводом результатов (ответ)
- Исходный код в приложении
- Ссылку на исходный код www.db-fiddle.com в приложении
- Ссылка на PR в приложении
- Вывод

Вариант 21

Пусть требуется создать программную систему, предназначенную для продавца журналов/комиксов. Такая система должна обеспечивать хранение сведений об имеющихся в магазине журналах, о читателях журналов и списке магазинов. Для каждого журнала в БД должны храниться следующие сведения: название журнала, серия, автор (ы), издательство, год издания, число экземпляров в каждом магазине, а также ISBN и дата продажи журнала. Сведения о читателях библиотеки должны включать почту (email), ФИО покупателя, дату рождения, адрес, номер телефон. Нужно учесть, что покупатели могут делать заказы с разных магазинов, но нужно сохранять информацию, о предпочтительных магазинах (в которых чаще делают заказы). Магазин имеет несколько отделов, которые характеризуются номером, названием и кол-во журналов (вместимость). Магазин может получать новые журналы и списывать старые. ISBN может измениться в результате переклассификации, а почта в результате перерегистрации. Продавцу могут потребоваться следующие сведения о текущем состоянии библиотеки:

- Какие журналы были куплены определенным покупателем?
- Как называется журнал с заданным ISBN?
- Какой ISBN у журнала с заданным названием?
- Когда журнал был куплен?
- Кто из покупателей купил журнал более месяца тому назад?
- Найти покупателя самых редких журналов (по наличию в магазине)?
- Какое число покупателей пользуется определенным магазином?
- Сколько покупателей младше 20 лет?

Выполнение работы.

В лабораторной работе №1 была разработана следующая реляционная модель:

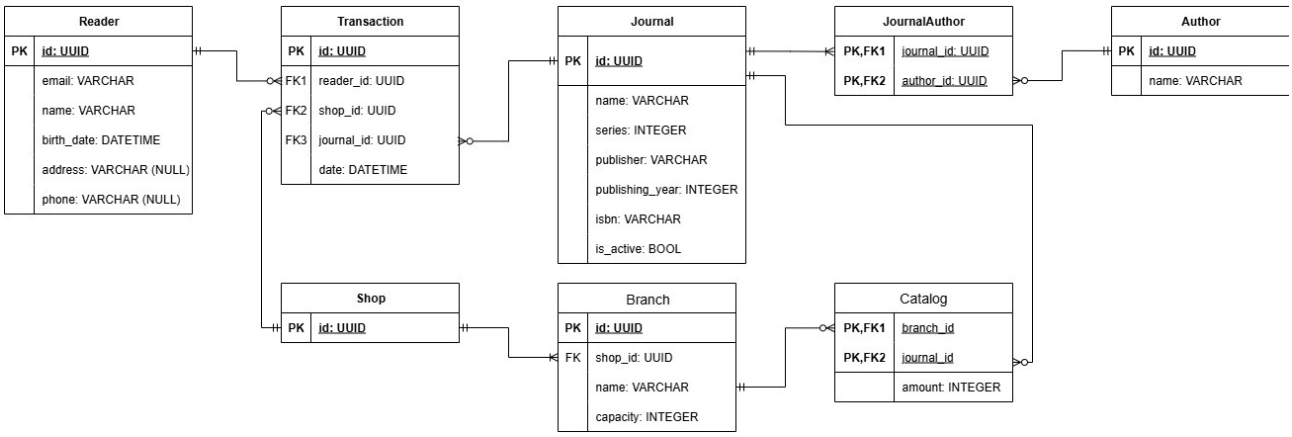

Рис. 1 — Структура базы данных

Для взаимодействия с СУБД PostgreSQL в работе использовался CLI интерфейс SQL Shell (psql). Была создана новая база данных для магазина комиксов при помощи команды *CREATE DATABASE comic_book_shop*.

```
postgres=# \l
                                                  Список баз данных
      Имя       | Владелец | Кодировка | Провайдер локали |     LC_COLLATE      |      LC_CTYPE       | Локаль | Правила ICU |     Права доступа
----------------+----------+-----------+------------------+---------------------+---------------------+--------+-------------+-----------------------
 comic_book_shop | postgres | UTF8      | libc             | Russian_Russia.1251 | Russian_Russia.1251 |        |             |
```

Рис. 2 — Работа с СУБД PostgreSQL

Далее с помощью команды *\c comic_book_shop* происходит подключение к базе данных. С помощью команды *CREATE TABLE* были созданы таблицы *author*, *branch*, *catalog*, *journal*, *journal_author*, *reader*, *shop*, *transaction*. Для обеспечения целостности и связности данных были определены первичные и внешние ключи, а также ограничения уникальности для полей вроде *email* читателя полиса и правила ссылочной целостности. Затем созданные таблицы были заполнены соответствующими данными с помощью команды *INSERT INTO*. Чтобы убедиться в успешности действий, был выполнен следующий запрос (см. рис. 3). Можем видеть, что таблица действительно заполнена требуемыми данными.

```
comic_book_shop=# SELECT * FROM reader;
                  id                  |          email          |      name       | birth_date |                address                 |      phone
--------------------------------------+-------------------------+-----------------+------------+----------------------------------------+-----------------
 66c8c0fd-d849-43b4-ae97-5a7cf04a1d4e | ivan.petrov@email.com   | Иван Петров     | 1990-05-15 | г. Москва, ул. Ленина, д. 10, кв. 25   | +7-915-123-45-67
 5740d88c-701c-4137-a4ba-91f78e993467 | anna.sidorova@email.com | Анна Сидорова   | 1985-12-03 | г. Санкт-Петербург, Невский пр-т, д. 50 |
 9477efca-4b5e-418b-8983-22c70e74eb2b | sergey.kozlov@email.com | Сергей Козлов   | 1998-08-20 |                                        | +7-916-987-65-43
 5d10d4b5-9045-4aa0-956d-65c56b204140 | maria.ivanova@email.com | Мария Иванова   | 2010-03-10 |                                        |
 11309de7-2b40-4ee2-ae96-c891faf5a9d1 | dmitry.sokolov@email.com | Дмитрий Соколов | 2011-11-28 | г. Казань, ул. Баумана, д. 15          |
(5 строк)
```

Рис. 3 — Результат заполнения одной из таблиц

Для ответа на вопросы из предыдущей лабораторной работы были написаны следующие запросы:

- Какие журналы были куплены определенным покупателем?

Запрос обращается к таблице transaction, к ней присоединяются (*JOIN*) таблицы *reader* и *journal*, база данных сопоставляет строки из таблиц по полям *reader_id* и *journal_id* с *id reader* и *journal* соответственно. После объединения данных с помощью команды *WHERE* из полученной таблицы отбираются только те строки, которые относятся к покупателю с указанным *email*. Выборка столбцов с *email* и названием журнала осуществляется командой *SELECT*.

```
comic_book_shop=# SELECT
comic_book_shop-# r.email AS reader,
comic_book_shop-# j.name AS journal
comic_book_shop-# FROM transaction t
comic_book_shop-# JOIN reader r ON t.reader_id=r.id
comic_book_shop-# JOIN journal j ON t.journal_id=j.id
comic_book_shop-# WHERE r.email='ivan.petrov@email.com';
        reader         |         journal
-----------------------+-------------------------
 ivan.petrov@email.com | The Amazing Spider-Man
 ivan.petrov@email.com | X-Men
(2 строки)
```

Рис. 4 — Результат первого запроса

- Как называется журнал с заданным ISBN?

В таблице *journals* ищутся строки с нужным *isbn*. Выводится *isbn* и соответствующий ему журнал.

```
comic_book_shop=# SELECT
comic_book_shop-# isbn,
comic_book_shop-# name AS journal
comic_book_shop-# FROM journal
comic_book_shop-# WHERE isbn='978-1-302-95001-1';
       isbn         |        journal
--------------------+-----------------------
 978-1-302-95001-1 | The Amazing Spider-Man
(1 строка)
```

Рис. 5 — Результат второго запроса

- Какой ISBN у журнала с заданным названием?

  Аналогично предыдущему запросу формируется и этот.

```
comic_book_shop=# SELECT
comic_book_shop-# name AS journal,
comic_book_shop-# isbn
comic_book_shop-# FROM journal
comic_book_shop-# WHERE name='Teen Titans Go!';
     journal      |       isbn
------------------+-------------------
 Teen Titans Go! | 978-1-7795-1789-1
(1 строка)
```

Рис. 6 — Результат третьего запроса

- Когда журнал был куплен?

К таблице *transaction* присоединяется (*JOIN*) по полю *journal_id* таблица *journal*. Выводятся строки, соответствующие выбранному журналу, отображается название и дата покупки журнала.

```
comic_book_shop=# SELECT
comic_book_shop-# j.name AS journal,
comic_book_shop-# t.date
comic_book_shop-# FROM transaction t
comic_book_shop-# JOIN journal j ON t.journal_id=j.id
comic_book_shop-# WHERE j.name='Teen Titans Go!';
     journal     |          date
-----------------+------------------------
 Teen Titans Go! | 2024-01-21 12:40:00+03
(1 строка)
```

Рис. 7 — Результат четвёртого запроса

- Кто из покупателей купил журнал более месяца тому назад?

Данные из таблицы *transaction* соединяются с данными из *reader* по *reader_id* и с *journal* по *journal_id*. Из всех строк выбираются те, в которых промежуток времени между *date* и текущей датой (*AGE*) больше 1 месяца (*INTERVAL '1 month'*). Выбирается столбец с датой транзакции, названием журнала и *email* читателя.

```
comic_book_shop=# SELECT
comic_book_shop-# t.date,
comic_book_shop-# j.name AS journal,
comic_book_shop-# r.email AS reader
comic_book_shop-# FROM transaction t
comic_book_shop-# JOIN reader r ON t.reader_id=r.id
comic_book_shop-# JOIN journal j ON t.journal_id=j.id
comic_book_shop-# WHERE AGE(t.date) > INTERVAL '1 month';
         date          |          journal          |          reader
-----------------------+---------------------------+-------------------------
 2024-01-15 10:30:00+03 | The Amazing Spider-Man    | ivan.petrov@email.com
 2024-01-16 14:20:00+03 | Batman: The Dark Knight   | anna.sidorova@email.com
 2024-01-17 16:45:00+03 | Attack on Titan           | sergey.kozlov@email.com
 2024-01-18 11:15:00+03 | The Sandman: Overture     | maria.ivanova@email.com
 2024-01-19 13:30:00+03 | Spider-Gwen: Ghost-Spider | dmitry.sokolov@email.com
 2024-01-20 15:20:00+03 | X-Men                     | ivan.petrov@email.com
 2024-01-21 12:40:00+03 | Teen Titans Go!           | anna.sidorova@email.com
 2024-01-22 17:10:00+03 | My Hero Academia          | sergey.kozlov@email.com
 2024-01-23 09:50:00+03 | X-Men                     | maria.ivanova@email.com
 2024-01-24 18:30:00+03 | The Amazing Spider-Man    | dmitry.sokolov@email.com
(10 строк)
```

Рис. 8 — Результат пятого запроса

- Найти покупателя самых редких журналов (по наличию в магазине)?

Для начала формируется временная таблица *rare_journals* (*WITH*). Ее столбцы *id* и *name* от журнала, а также *copies*, который вычисляется как сумма (*SUM*) экземпляров журнала во всех каталогах. Для этого присоединяется таблица catalog, причем слева. Таким образом, если журнала нет в каталоге, его количество будет *NULL*. Поэтому необходим *COALESCE*, чтобы преобразовать *NULL* в 0. После строки группируются по *id* и *name*. Дальше оставляются только те, где количество журналов больше 0 (*HAVING*). Строки сортируются по возрастанию копий (*ORDER BY*, *ASC*), оставляются только 3 самых редких журналов (*LIMIT 3*). После, как при обычном поиске покупателя журнала, через таблицу *transaction* происходит сопоставление. Выводятся *email* покупателя, имя и количество копий редких журналов.

```
comic_book_shop=# WITH rare_journal AS (
comic_book_shop(#   SELECT
comic_book_shop(#   j.id,
comic_book_shop(#   j.name,
comic_book_shop(#   COALESCE(SUM(c.amount),0) as copies
comic_book_shop(#   FROM journal j
comic_book_shop(#   LEFT JOIN catalog c ON j.id=c.journal_id
comic_book_shop(#   GROUP BY j.id, j.name
comic_book_shop(#   HAVING COALESCE(SUM(c.amount),0) > 0
comic_book_shop(#   ORDER BY copies ASC
comic_book_shop(#   LIMIT 3
comic_book_shop(# )
comic_book_shop-#
comic_book_shop-# SELECT
comic_book_shop-# r.email AS reader,
comic_book_shop-# rj.name AS rare_journal,
comic_book_shop-# rj.copies
comic_book_shop-# FROM transaction t
comic_book_shop-# JOIN reader r ON t.reader_id=r.id
comic_book_shop-# JOIN rare_journal rj ON t.journal_id=rj.id;
         reader          |      rare_journal       | copies
-------------------------+-------------------------+--------
 anna.sidorova@email.com | Batman: The Dark Knight |      8
 maria.ivanova@email.com | The Sandman: Overture   |      5
 anna.sidorova@email.com | Teen Titans Go!         |     10
(3 строки)
```

Рис. 9 — Результат шестого запроса

- Какое число покупателей пользуется определенным магазином?

Объединяются таблицы *transaction*, *reader*, *shop*. Подсчитывается (*COUNT*) количество различных (*DISTINCT*) покупателей.

```
comic_book_shop=# SELECT
comic_book_shop-# COUNT(DISTINCT r.id) AS readers
comic_book_shop-# FROM transaction t
comic_book_shop-# JOIN reader r ON t.reader_id=r.id
comic_book_shop-# JOIN shop s ON t.shop_id=s.id
comic_book_shop-# WHERE s.id='4db426cb-00e3-477c-9f78-9eaf82c968b3';
 readers
---------
       3
(1 строка)
```

Рис. 10 — Результат седьмого запроса

- Сколько покупателей младше 20 лет?

В таблице reader подсчитывается количество строк, где промежуток времени между *birth_date* и текущей датой (*AGE*) меньше 20 лет (*INTERVAL '20 years'*).

```
comic_book_shop=# SELECT
comic_book_shop-# COUNT(DISTINCT id) AS readers
comic_book_shop-# FROM reader
comic_book_shop-# WHERE AGE(birth_date) < INTERVAL '20 years';
 readers
---------
       2
(1 строка)
```

Рис. 11 — Результат восьмого запроса

Написанный программный код см. в приложении А.

Выводы.

В результате выполнения лабораторной работы была развёрнута база данных в СУБД PostgreSQL с использованием CLI интерфейса SQL Shell (psql), соответствующая разработанной ранее модели. Были выполнены запросы для создания необходимых таблиц и извлечения данных из этих таблиц.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Ссылка на Pull-request с результатами выполненной лабораторной работы: https://github.com/moevm/sql-2025-3341/pull/18

Исходный код: https://www.db-fiddle.com/f/8CH9XjYWCqWxbmaUMqroR9/6

Файл lab2.sql:

```
CREATE TABLE public.author (
    id uuid DEFAULT gen_random_uuid() NOT NULL,
    name character varying NOT NULL
);


CREATE TABLE public.branch (
    id uuid DEFAULT gen_random_uuid() NOT NULL,
    shop_id uuid NOT NULL,
    name character varying NOT NULL,
    capacity integer NOT NULL
);


CREATE TABLE public.catalog (
    branch_id uuid NOT NULL,
    journal_id uuid NOT NULL,
    amount integer NOT NULL
);


CREATE TABLE public.journal (
    id uuid DEFAULT gen_random_uuid() NOT NULL,
    name character varying NOT NULL,
    series integer NOT NULL,
```

```
    publisher character varying NOT NULL,
    publishing_year integer NOT NULL,
    isbn character varying NOT NULL,
    is_active boolean NOT NULL
);


CREATE TABLE public.journal_author (
    journal_id uuid NOT NULL,
    author_id uuid NOT NULL
);


CREATE TABLE public.reader (
    id uuid DEFAULT gen_random_uuid() NOT NULL,
    email character varying NOT NULL,
    name character varying NOT NULL,
    birth_date date NOT NULL,
    address character varying,
    phone character varying
);


CREATE TABLE public.shop (
    id uuid DEFAULT gen_random_uuid() NOT NULL
);


CREATE TABLE public.transaction (
    id uuid DEFAULT gen_random_uuid() NOT NULL,
    reader_id uuid NOT NULL,
    shop_id uuid NOT NULL,
    journal_id uuid NOT NULL,
    date timestamp with time zone NOT NULL
);
```

```
    INSERT INTO public.author VALUES ('737fbeb0-a0c4-4df1-a396-e4ae45377e86', 'Stan Lee');
    INSERT INTO public.author VALUES ('008686be-9460-4b34-992c-4f2d2d75c44a', 'Bob Kane');
    INSERT INTO public.author VALUES ('fcd98444-756e-4fff-a7ad-2f481f142994', 'Jerry Siegel');
    INSERT INTO public.author VALUES ('130d813f-6f13-4d6d-8a09-8fc1475658ea', 'Hajime Isayama');
    INSERT INTO public.author VALUES ('dd4b104b-df47-4637-a3f2-a3db44418dfa', 'Kohei Horikoshi');
    INSERT INTO public.author VALUES ('5a4a8e0b-95a5-4d21-be43-c8494fabd4e3', 'Neil Gaiman');
    INSERT INTO public.author VALUES ('a8d3940a-0a00-4ab5-8af5-44b6ed46c999', 'Jason Latour');
    INSERT INTO public.author VALUES ('e01519a1-606f-4f2b-a6cd-9eb684ac1e8d', 'J. Torres');


    INSERT INTO public.branch VALUES ('a025e999-5b29-4a61-892a-201b3ff173aa', '4db426cb-00e3-477c-9f78-9eaf82c968b3', 'Комиксы Marvel', 50);
    INSERT INTO public.branch VALUES ('1448d45d-cf20-4fc7-a53e-7a455a7b8d1c', '4db426cb-00e3-477c-9f78-9eaf82c968b3', 'Комиксы DC', 45);
    INSERT INTO public.branch VALUES ('ca339d8a-4dd8-4cdb-85b1-f7abc942f6a1', 'a8029dce-c70b-43a7-bb69-0829d1d41bca', 'Манга и аниме', 60);
    INSERT INTO public.branch VALUES ('9bc65a0e-4deb-4952-98a3-aadec14fd6af', 'a8029dce-c70b-43a7-bb69-0829d1d41bca', 'Графические романы', 35);
    INSERT INTO public.branch VALUES ('787f74cf-4583-44eb-88cf-ac20433017a3', 'cd707836-60d4-4213-910d-eb6dad43fe86', 'Новинки', 40);
```

```
    INSERT INTO public.branch VALUES ('b415ce4b-0d5c-4e59-8920-68c39e862b4e', 'cd707836-60d4-4213-910d-eb6dad43fe86', 'Раритетные издания', 25);
    INSERT INTO public.branch VALUES ('7caf2960-6006-40c9-9455-a8d91a01f1b9', '7e0f0fc4-fcf4-4bcb-a415-e0824f1b6412', 'Детские комиксы', 55);
    INSERT INTO public.branch VALUES ('fc6b205a-3d6e-4862-baa0-1c5962fca291', '7e0f0fc4-fcf4-4bcb-a415-e0824f1b6412', 'Западные комиксы', 48);
    INSERT INTO public.branch VALUES ('00943eb3-5a06-4f2b-93de-7f82ae5732a8', 'dba0ed7e-d2fc-47d7-be1e-db8a3a9d0476', 'Супергерои', 52);
    INSERT INTO public.branch VALUES ('749ced6a-5a79-46bc-984e-98f5abe5bfd4', 'dba0ed7e-d2fc-47d7-be1e-db8a3a9d0476', 'Научная фантастика', 38);


    INSERT INTO public.catalog VALUES ('a025e999-5b29-4a61-892a-201b3ff173aa', '0bc82198-02a1-4228-84fa-2c48f4c7c793', 12);
    INSERT INTO public.catalog VALUES ('a025e999-5b29-4a61-892a-201b3ff173aa', 'ef947889-d288-4ded-86d2-5f26d46a9c85', 10);
    INSERT INTO public.catalog VALUES ('1448d45d-cf20-4fc7-a53e-7a455a7b8d1c', 'f5751297-75b7-4283-a3e8-f4bd0a84168c', 8);
    INSERT INTO public.catalog VALUES ('ca339d8a-4dd8-4cdb-85b1-f7abc942f6a1', '0045dc9a-8cef-4e17-976a-82a8b26ca9a4', 15);
    INSERT INTO public.catalog VALUES ('ca339d8a-4dd8-4cdb-85b1-f7abc942f6a1', 'eb7446ff-4ee3-410b-b41a-8d5ed0521cc6', 12);
    INSERT INTO public.catalog VALUES ('9bc65a0e-4deb-4952-98a3-aadec14fd6af', '655f827b-c785-4b43-979d-47b6f11d31d4', 5);
    INSERT INTO public.catalog VALUES ('787f74cf-4583-44eb-88cf-ac20433017a3', '664969a2-0804-4390-9400-7c24f8eb2066', 8);
    INSERT INTO public.catalog VALUES ('7caf2960-6006-40c9-9455-a8d91a01f1b9', '1bb5c6ac-b86a-4d33-9e0a-f6a3e3dce2d5', 10);
    INSERT INTO public.catalog VALUES ('fc6b205a-3d6e-4862-baa0-1c5962fca291', 'ef947889-d288-4ded-86d2-5f26d46a9c85', 7);
```

```
    INSERT INTO public.catalog VALUES ('00943eb3-5a06-4f2b-93de-7f82ae5732a8', '664969a2-0804-4390-9400-7c24f8eb2066', 6);


    INSERT INTO public.journal VALUES ('0bc82198-02a1-4228-84fa-2c48f4c7c793', 'The Amazing Spider-Man', 1, 'Marvel', 2024, '978-1-302-95001-1', true);
    INSERT INTO public.journal VALUES ('ef947889-d288-4ded-86d2-5f26d46a9c85', 'X-Men', 7, 'Marvel', 2023, '978-1-302-93456-1', true);
    INSERT INTO public.journal VALUES ('f5751297-75b7-4283-a3e8-f4bd0a84168c', 'Batman: The Dark Knight', 1, 'DC Comics', 2024, '978-1-7795-1651-1', true);
    INSERT INTO public.journal VALUES ('d30d33f8-c0d9-4078-a4c0-86b67be587bb', 'Superman: Legacy', 3, 'DC Comics', 2023, '978-1-7795-1287-2', false);
    INSERT INTO public.journal VALUES ('0045dc9a-8cef-4e17-976a-82a8b26ca9a4', 'Attack on Titan', 34, 'Kodansha', 2023, '978-1-63236-978-1', true);
    INSERT INTO public.journal VALUES ('eb7446ff-4ee3-410b-b41a-8d5ed0521cc6', 'My Hero Academia', 32, 'Shueisha', 2024, '978-1-9747-2345-6', true);
    INSERT INTO public.journal VALUES ('655f827b-c785-4b43-979d-47b6f11d31d4', 'The Sandman: Overture', 1, 'Vertigo', 2023, '978-1-4012-6552-1', true);
    INSERT INTO public.journal VALUES ('664969a2-0804-4390-9400-7c24f8eb2066', 'Spider-Gwen: Ghost-Spider', 1, 'Marvel', 2024, '978-1-302-95678-1', true);
    INSERT INTO public.journal VALUES ('3497d265-40a6-43fc-8858-01a8e555fa5e', 'The Fantastic Four', 1, 'Marvel', 1961, '978-1-302-90001-1', false);
    INSERT INTO public.journal VALUES ('1bb5c6ac-b86a-4d33-9e0a-f6a3e3dce2d5', 'Teen Titans Go!', 15, 'DC Comics', 2024, '978-1-7795-1789-1', true);
```

```
    INSERT INTO public.journal_author VALUES ('0bc82198-02a1-4228-84fa-2c48f4c7c793', '737fbeb0-a0c4-4df1-a396-e4ae45377e86');
    INSERT INTO public.journal_author VALUES ('ef947889-d288-4ded-86d2-5f26d46a9c85', '737fbeb0-a0c4-4df1-a396-e4ae45377e86');
    INSERT INTO public.journal_author VALUES ('3497d265-40a6-43fc-8858-01a8e555fa5e', '737fbeb0-a0c4-4df1-a396-e4ae45377e86');
    INSERT INTO public.journal_author VALUES ('f5751297-75b7-4283-a3e8-f4bd0a84168c', '008686be-9460-4b34-992c-4f2d2d75c44a');
    INSERT INTO public.journal_author VALUES ('d30d33f8-c0d9-4078-a4c0-86b67be587bb', 'fcd98444-756e-4fff-a7ad-2f481f142994');
    INSERT INTO public.journal_author VALUES ('0045dc9a-8cef-4e17-976a-82a8b26ca9a4', '130d813f-6f13-4d6d-8a09-8fc1475658ea');
    INSERT INTO public.journal_author VALUES ('eb7446ff-4ee3-410b-b41a-8d5ed0521cc6', 'dd4b104b-df47-4637-a3f2-a3db44418dfa');
    INSERT INTO public.journal_author VALUES ('655f827b-c785-4b43-979d-47b6f11d31d4', '5a4a8e0b-95a5-4d21-be43-c8494fabd4e3');
    INSERT INTO public.journal_author VALUES ('664969a2-0804-4390-9400-7c24f8eb2066', 'a8d3940a-0a00-4ab5-8af5-44b6ed46c999');
    INSERT INTO public.journal_author VALUES ('1bb5c6ac-b86a-4d33-9e0a-f6a3e3dce2d5', 'e01519a1-606f-4f2b-a6cd-9eb684ac1e8d');


    INSERT INTO public.reader VALUES ('66c8c0fd-d849-43b4-ae97-5a7cf04a1d4e', 'ivan.petrov@email.com', 'Иван Петров', '1990-05-15', 'г. Москва, ул. Ленина, д. 10, кв. 25', '+7-915-123-45-67');
    INSERT INTO public.reader VALUES ('5740d88c-701c-4137-a4ba-91f78e993467', 'anna.sidorova@email.com', 'Анна Сидорова', '1985-12-03', 'г. Санкт-Петербург, Невский пр-т, д. 50', NULL);
    INSERT INTO public.reader VALUES ('9477efca-4b5e-418b-8983-22c70e74eb2b', 'sergey.kozlov@email.com', 'Сергей Козлов', '1998-08-20', NULL, '+7-916-987-65-43');
    INSERT INTO public.reader VALUES ('5d10d4b5-9045-4aa0-956d-65c56b204140', 'maria.ivanova@email.com', 'Мария Иванова', '2010-03-10', NULL, NULL);
```

```
    INSERT INTO public.reader VALUES ('11309de7-2b40-4ee2-ae96-
c891faf5a9d1', 'dmitry.sokolov@email.com', 'Дмитрий Соколов',
'2011-11-28', 'г. Казань, ул. Баумана, д. 15', NULL);


    INSERT INTO public.shop VALUES ('4db426cb-00e3-477c-9f78-
9eaf82c968b3');
    INSERT INTO public.shop VALUES ('a8029dce-c70b-43a7-bb69-
0829d1d41bca');
    INSERT INTO public.shop VALUES ('cd707836-60d4-4213-910d-
eb6dad43fe86');
    INSERT INTO public.shop VALUES ('7e0f0fc4-fcf4-4bcb-a415-
e0824f1b6412');
    INSERT INTO public.shop VALUES ('dba0ed7e-d2fc-47d7-be1e-
db8a3a9d0476');


    INSERT INTO public.transaction VALUES ('9a34773e-2d46-4a73-
b099-ee745a62a3f6',        '66c8c0fd-d849-43b4-ae97-5a7cf04a1d4e',
'4db426cb-00e3-477c-9f78-9eaf82c968b3', '0bc82198-02a1-4228-84fa-
2c48f4c7c793', '2024-01-15 10:30:00+03');
    INSERT INTO public.transaction VALUES ('c163b0ca-cd16-4585-
ad0e-dc831927b090',        '5740d88c-701c-4137-a4ba-91f78e993467',
'4db426cb-00e3-477c-9f78-9eaf82c968b3', 'f5751297-75b7-4283-a3e8-
f4bd0a84168c', '2024-01-16 14:20:00+03');
    INSERT INTO public.transaction VALUES ('c1f985bc-71cb-4719-
bc64-c01ebc6d48d9',        '9477efca-4b5e-418b-8983-22c70e74eb2b',
'a8029dce-c70b-43a7-bb69-0829d1d41bca', '0045dc9a-8cef-4e17-976a-
82a8b26ca9a4', '2024-01-17 16:45:00+03');
    INSERT INTO public.transaction VALUES ('cabb4e87-f218-40f3-
82ca-6b27b179c9ae',        '5d10d4b5-9045-4aa0-956d-65c56b204140',
'a8029dce-c70b-43a7-bb69-0829d1d41bca', '655f827b-c785-4b43-979d-
47b6f11d31d4', '2024-01-18 11:15:00+03');
    INSERT INTO public.transaction VALUES ('008a3ef4-9786-4f28-
9731-b581045cf338',        '11309de7-2b40-4ee2-ae96-c891faf5a9d1',
```

```
'cd707836-60d4-4213-910d-eb6dad43fe86',   '664969a2-0804-4390-9400-
7c24f8eb2066', '2024-01-19 13:30:00+03');
    INSERT INTO public.transaction VALUES ('5404b7d2-989d-4241-
933e-3fe6492c291d',        '66c8c0fd-d849-43b4-ae97-5a7cf04a1d4e',
'7e0f0fc4-fcf4-4bcb-a415-e0824f1b6412',   'ef947889-d288-4ded-86d2-
5f26d46a9c85', '2024-01-20 15:20:00+03');
    INSERT INTO public.transaction VALUES ('c2fb3ebd-361b-40a0-
b060-bff05282bb84',        '5740d88c-701c-4137-a4ba-91f78e993467',
'7e0f0fc4-fcf4-4bcb-a415-e0824f1b6412',   '1bb5c6ac-b86a-4d33-9e0a-
f6a3e3dce2d5', '2024-01-21 12:40:00+03');
    INSERT INTO public.transaction VALUES ('0f8aaeb5-8470-4723-
97b9-3371f2f9cbea',        '9477efca-4b5e-418b-8983-22c70e74eb2b',
'a8029dce-c70b-43a7-bb69-0829d1d41bca',   'eb7446ff-4ee3-410b-b41a-
8d5ed0521cc6', '2024-01-22 17:10:00+03');
    INSERT INTO public.transaction VALUES ('00a0bd29-88ee-4cbc-
8503-76f9f2ea6499',        '5d10d4b5-9045-4aa0-956d-65c56b204140',
'dba0ed7e-d2fc-47d7-be1e-db8a3a9d0476',   'ef947889-d288-4ded-86d2-
5f26d46a9c85', '2024-01-23 09:50:00+03');
    INSERT INTO public.transaction VALUES ('47e71442-9308-402c-
8bd9-a4c1c1b8b786',        '11309de7-2b40-4ee2-ae96-c891faf5a9d1',
'4db426cb-00e3-477c-9f78-9eaf82c968b3',   '0bc82198-02a1-4228-84fa-
2c48f4c7c793', '2024-01-24 18:30:00+03');


    ALTER TABLE ONLY public.author
        ADD CONSTRAINT author_pkey PRIMARY KEY (id);


    ALTER TABLE ONLY public.branch
        ADD CONSTRAINT branch_pkey PRIMARY KEY (id);


    ALTER TABLE ONLY public.catalog
        ADD  CONSTRAINT  catalog_pkey  PRIMARY  KEY  (branch_id,
journal_id);
```

```
    ALTER TABLE ONLY public.journal_author
        ADD CONSTRAINT journal_author_pkey PRIMARY KEY
(journal_id, author_id);


    ALTER TABLE ONLY public.journal
        ADD CONSTRAINT journal_pkey PRIMARY KEY (id);


    ALTER TABLE ONLY public.reader
        ADD CONSTRAINT reader_email_key UNIQUE (email);


    ALTER TABLE ONLY public.reader
        ADD CONSTRAINT reader_pkey PRIMARY KEY (id);


    ALTER TABLE ONLY public.shop
        ADD CONSTRAINT shop_pkey PRIMARY KEY (id);


    ALTER TABLE ONLY public.transaction
        ADD CONSTRAINT transaction_pkey PRIMARY KEY (id);


    CREATE INDEX idx_catalog_amount ON public.catalog USING btree
(amount);


    CREATE INDEX idx_journal_isbn ON public.journal USING btree
(isbn);
```

```
    CREATE INDEX idx_journal_name ON public.journal USING btree
(name);


    CREATE INDEX idx_reader_birth_date ON public.reader USING
btree (birth_date);


    CREATE INDEX idx_transaction_date ON public.transaction USING
btree (date);


    CREATE INDEX idx_transaction_journal_id ON public.transaction
USING btree (journal_id);


    CREATE INDEX idx_transaction_reader_id ON public.transaction
USING btree (reader_id);


    CREATE INDEX idx_transaction_shop_id ON public.transaction
USING btree (shop_id);


    ALTER TABLE ONLY public.branch
        ADD CONSTRAINT branch_shop_id_fkey FOREIGN KEY (shop_id)
REFERENCES public.shop(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.catalog
        ADD CONSTRAINT catalog_branch_id_fkey FOREIGN KEY
(branch_id) REFERENCES public.branch(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.catalog
```

```
        ADD CONSTRAINT catalog_journal_id_fkey FOREIGN KEY
(journal_id) REFERENCES public.journal(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.journal_author
        ADD CONSTRAINT journal_author_author_id_fkey FOREIGN KEY
(author_id) REFERENCES public.author(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.journal_author
        ADD CONSTRAINT journal_author_journal_id_fkey FOREIGN KEY
(journal_id) REFERENCES public.journal(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.transaction
        ADD CONSTRAINT transaction_journal_id_fkey FOREIGN KEY
(journal_id) REFERENCES public.journal(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.transaction
        ADD CONSTRAINT transaction_reader_id_fkey FOREIGN KEY
(reader_id) REFERENCES public.reader(id) ON DELETE RESTRICT;


    ALTER TABLE ONLY public.transaction
        ADD CONSTRAINT transaction_shop_id_fkey FOREIGN KEY
(shop_id) REFERENCES public.shop(id) ON DELETE RESTRICT;


    -- Какие журналы были куплены определенным покупателем?
    -- email уникален, так что его можно использовать как ключ
    -- Можно также использовать id читателя, но email проще :)
    SELECT
    r.email AS reader,
    j.name AS journal
```

```
FROM transaction t
JOIN reader r ON t.reader_id=r.id
JOIN journal j ON t.journal_id=j.id
WHERE r.email='ivan.petrov@email.com';


-- Как называется журнал с заданным ISBN?
SELECT
isbn,
name AS journal
FROM journal
WHERE isbn='978-1-302-95001-1';


-- Какой ISBN у журнала с заданным названием?
-- В дальнейшем имя журнала считается уникальным, хотя это может быть не так
-- Лучше использовать id, но он длинный :(
SELECT
name AS journal,
isbn
FROM journal
WHERE name='Teen Titans Go!';


-- Когда журнал был куплен?
SELECT
j.name AS journal,
t.date
FROM transaction t
JOIN journal j ON t.journal_id=j.id
WHERE j.name='Teen Titans Go!';


-- Кто из покупателей купил журнал более месяца тому назад?
```

```
SELECT
t.date,
j.name AS journal,
r.email AS reader
FROM transaction t
JOIN reader r ON t.reader_id=r.id
JOIN journal j ON t.journal_id=j.id
WHERE AGE(t.date) > INTERVAL '1 month';


-- Найти покупателя самых редких журналов (по наличию в магазине)?
WITH rare_journal AS (
  SELECT
  j.id,
  j.name,
  COALESCE(SUM(c.amount),0) as copies
  FROM journal j
  LEFT JOIN catalog c ON j.id=c.journal_id
  GROUP BY j.id, j.name
  HAVING COALESCE(SUM(c.amount),0) > 0
  ORDER BY copies ASC
  LIMIT 3
)
SELECT
r.email AS reader,
rj.name AS rare_journal,
rj.copies
FROM transaction t
JOIN reader r ON t.reader_id=r.id
JOIN rare_journal rj ON t.journal_id=rj.id;


-- Какое число покупателей пользуется определенным магазином?
SELECT
```

```
    COUNT(DISTINCT r.id) AS readers
    FROM transaction t
    JOIN reader r ON t.reader_id=r.id
    JOIN shop s ON t.shop_id=s.id
    WHERE s.id='4db426cb-00e3-477c-9f78-9eaf82c968b3';


    -- Сколько покупателей младше 20 лет?
    SELECT
    COUNT(DISTINCT id) AS readers
    FROM reader
WHERE AGE(birth_date) < INTERVAL '20 years';
```