Используется СУБД SQLLITE, так как данная СУБД отлично совместима с python, а также не требует установки сервера, но при этом обладает всем необходимым функционалом для написания запросов из ТЗ.

Задание 1

Дана модель фрагмента БД Библиотека:

Книги:
- ID книги
- Название
- Автор
- Издательство
- Год издания
- Город издания
- Количество страниц

Экземпляры книг:
- ID экземпляра
- ID книги

Выдачи книг:
- ID экземпляра
- Дата выдачи
- Дата возврата
- № читательского билета

Читатели:
- № читательского билета
- Фамилия
- Имя
- Отчество
- Дата рождения
- Пол
- Адрес
- Телефон

Необходимо написать следующие запросы к БД:

1. Найти город (или города), в котором в 2016 году было издано больше всего книг (не экземпляров).
2. Вывести количество экземпляров книг «Война и мир» Л.Н.Толстого, которые сейчас находятся в библиотеке (не на руках у читателей).
3. Найти читателя, который за последний месяц брал больше всего книг в библиотеке.

Если читателей с максимальным количество несколько - вывести только тех, у кого самый маленький возраст.

Для решения данной задачи были реализованы следующие функции на языке Python:

1) Функция *create_database_tables* – принимает в качестве входного параметра объект *cursor* создает базу данных *library* и таблицы *Books*, *BookInstances*, *Readers*, *BookLoans*;
2) Функции *add_random_book*, *add_random_reader*, *add_random_instance*, *add_random_book_loan* принимают качестве входных параметров объект *cursor* и количество строк (*count_rows*). Функции заполняют БД *count_rows* случайных строк, созданных при помощи библиотеки *Faker*;
3) Функция *request_1* выполняет первый запрос к БД:

```
SELECT publication_city, COUNT(*) as book_count
        FROM Books
        WHERE publication_year = 2016
        GROUP BY publication_city
        HAVING COUNT(*) = (
          SELECT MAX(book_count)
          FROM (
            SELECT COUNT(*) as book_count
            FROM Books
            WHERE publication_year = 2016
            GROUP BY publication_city
          ) AS subquery
        );
```

4) Функция *request_2* выполняет второй запрос к БД:

```
SELECT COUNT(book_count), author FROM
        (SELECT COUNT(BookInstances.instance_id) as book_count, Books.author as author
FROM BookInstances
JOIN BookLoans ON BookInstances.instance_id=BookLoans.instance_id
JOIN Books ON BookInstances.book_id=Books.book_id
WHERE BookLoans.return_date is NOT NULL AND
Books.author = 'Л.Н.Толстой' ANDBooks.title = 'Война и мир' GROUP BY
BookInstances.instance_id)
```

5) Функция *request_3* выполняет третий запрос к БД:

```
SELECT name, last_name,  MAX(date_of_birth) FROM
        (SELECT Readers.first_name as name, COUNT(*) as count_loan, Readers.date_of_birth,
Readers.last_name as last_name FROM Readers
         JOIN BookLoans ON Readers.reader_ticket_number = BookLoans.reader_ticket_number
         WHERE BookLoans.loan_date >= DATE('now', '-1 month')
         GROUP BY Readers.reader_ticket_number
         HAVING count_loan = (
```

```
    SELECT MAX(count_loan)
    FROM (
      SELECT Readers.first_name, COUNT(*) as count_loan FROM Readers JOIN BookLoans
       ON Readers.reader_ticket_number = BookLoans.reader_ticket_number
       WHERE BookLoans.loan_date >= DATE('now', '-1 month')
       GROUP BY Readers.reader_ticket_number
    )
))
```

Задание 2

Доработать ЛМД таким образом, чтобы можно было хранить данные о сотрудниках библиотеки, которые выдали книгу читателю.
Нарисовать ER-диаграмму получившейся ЛМД.

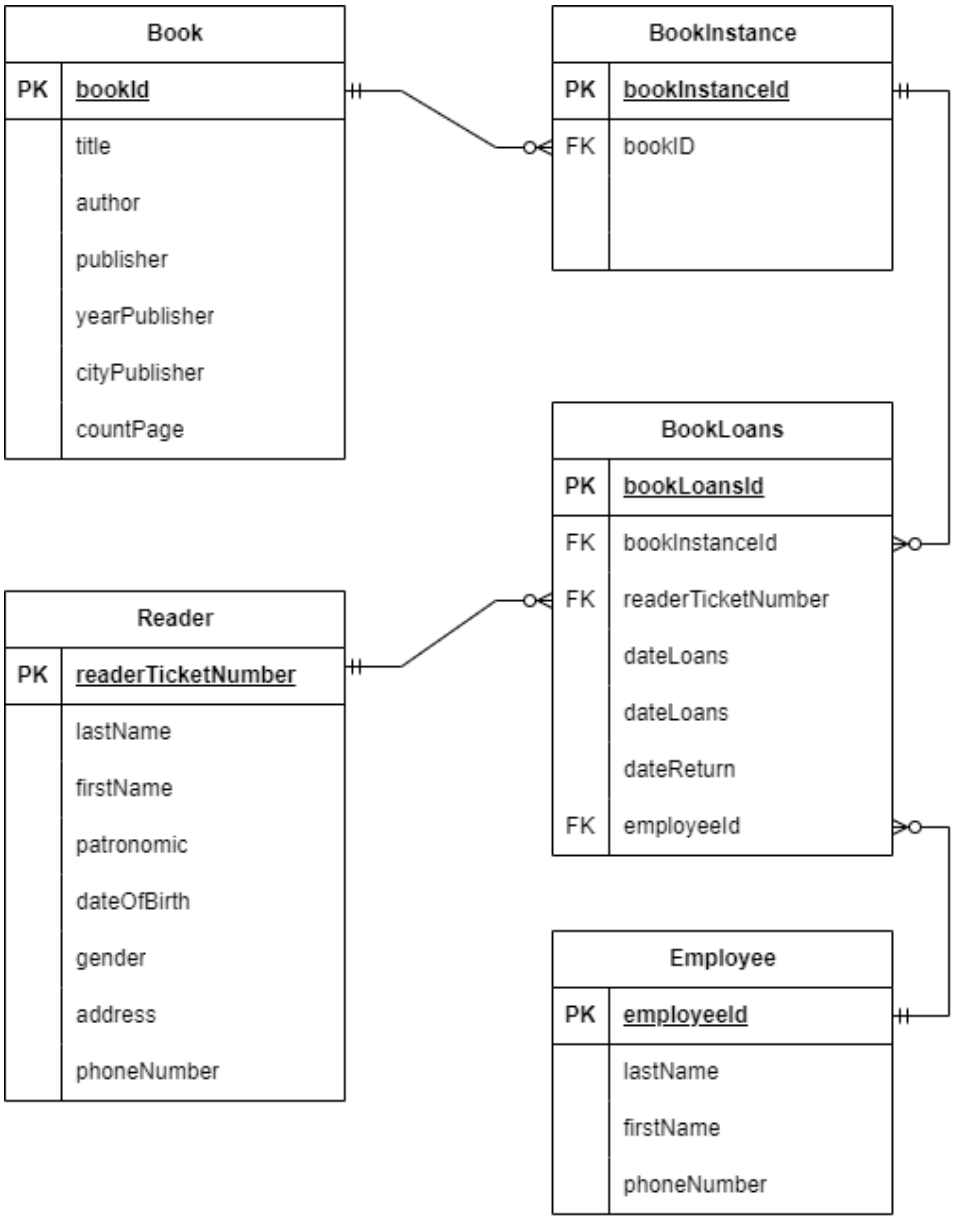

Рисунок 1. ER диаграмма.

Задание 3

Дан фрагмент таблицы электронного журнала:

- ФИО ученика
- Дата
- Оценка
- Предмет

1. Необходимо вывести ФИО ученика, у которого средняя оценка за всё время обучения > 4.5 и нет ни одной оценки 2 в этом календарном году.
2. Вывести для каждого ученика предмет с самой лучшей у него успеваемостью за весь период обучения (по средней оценке по предмету), и с самой худшей. Если предметов несколько, вывести первый по алфавитному порядку.

Для решения данной задачи были реализованы следующие функции на языке Python:

1) Функция *create_database_tables* – принимает в качестве входного параметра объект cursor создает базу данных *journal* и таблицу *Journal*;
2) Функция *add_random_journal* принимает в качестве входных параметров объект cursor и количество строк (*count_rows*). Функция заполняет БД *count_rows* случайных строк, созданных при помощи библиотеки *Faker*;
3) Функция *request_1* выполняет первый запрос к БД:

```
SELECT student_name, AVG(mark) as avg_mark FROM Journal
        WHERE student_name NOT IN (
            SELECT student_name
            FROM Journal
            WHERE mark = 2
              AND strftime('%Y', date) = strftime('%Y', 'now')
        )
        GROUP BY student_name
        HAVING avg_mark > 4.5
```

4) Функция *request_2* выполняет второй запрос к БД:

```
WITH AvgGrades AS
(SELECT student_name, subject, AVG(mark) as avg_mark,
RANK() OVER(PARTITION BY student_name ORDER BY AVG(mark) DESC) as rank_high,
RANK() OVER(PARTITION BY student_name ORDER BY AVG(mark) ASC) as rank_low
FROM Journal
GROUP BY student_name, subject)
SELECT student_name,
MAX(CASE WHEN rank_high = 1 THEN subject END) AS best_subject,
MAX(CASE WHEN rank_low = 1 THEN subject END) AS worst_subject
FROM AvgGrades
GROUP BY student_name
```

Задание 4

Дана модель фрагмента БД Колл-центра
Звонки
-ид клиента
-ид звонка
-дата и время звонка

Необходимо вывести ид клиента и кол-во сессий звонков этого клиента. Под сессиями понимаются звонки, между которыми прошло не более 30 минут. Например, клиент звонит в 11:00, 11:15, 11:17 и 13:20. Звонки в 11:00, 11:15 и 11:17 — это одна сессия. Звонок в 13:20 — это вторая сессия. Итого по этому клиенту было 2 сессии звонков.

Для решения данной задачи были реализованы следующие функции на языке Python:

1) Функция *create_database_tables* – принимает в качестве входного параметра объект cursor создает базу данных call_*centers* и таблицу *Calls*;
2) Функция *add_random_call* принимает в качестве входных параметров объект cursor и количество строк (*count_rows*). Функция заполняет БД *count_rows* случайных строк, созданных при помощи библиотеки *Faker*;

3) Функция *request_1* выполняет первый запрос к БД:

```
WITH CallsWithGaps AS (
SELECT id_client, id_call, call_datetime,
strftime('%s', call_datetime) - strftime('%s', lag(call_datetime) OVER (PARTITION BY id_client
ORDER BY call_datetime))
AS time_diff
FROM Calls),
CallsWithFlag AS (
SELECT id_client, time_diff, id_call,
(CASE WHEN time_diff < 3600 THEN 1 ELSE 0 END) as flag
FROM CallsWithGaps)
SELECT id_client, SUM(end_session) / 2 FROM(
SELECT
CASE WHEN lag(flag) OVER (PARTITION BY id_client) = flag THEN 0
ELSE 1 END as end_session,
flag,
id_client
FROM CallsWithFlag)
GROUP BY id_client
```