|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №4**

**«Использование БД а Andoid приложениях»**

**ДИСЦИПЛИНА: «Разработка мобильного ПО»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-52Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_Боков А.А.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_Прудяк П.Н.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2024 г. | | |

Цель: формирование практических навыков разработки приложений с использованием СУБД SQLite, списков и файлов при разработке Android-приложений с несколькими Activity.

Задачи:

1. Научиться работать с СУБД SQLite.

2. Научиться сохранять результаты выполнения запросов к базе

данных в списки, файлы и LogCat.

3. Понять особенности реализации Android-приложений с

использованием списков и СУБД SQLite

Формулировка задания:



**Листинг:**

**MainActivity.java**

package com.example.android\_dev\_lab4new;  
import static com.example.android\_dev\_lab4new.R.\*;  
  
import android.annotation.SuppressLint;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.os.Bundle;  
import android.text.InputType;  
import android.util.Log;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.LinearLayout;  
import android.widget.ListView;  
import android.widget.Spinner;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import java.io.BufferedReader;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class MainActivity extends AppCompatActivity {  
  
 private DBHelper dbHelper;  
 private TextView tvResults;  
 private Button btnSort, btnGroup, btnSum, btnAvg, btnMax, btnGreaterThan, btnLessThanAvg, btnTypeGreaterThan, btnReadFromFile, btnShowBooks;  
  
 private ListView lvResults;  
 private ArrayAdapter<String> adapter;  
 private ArrayList<String> resultList;  
  
 private void writeToFile(String data) {  
 try {  
 // Используем режим MODE\_APPEND, чтобы добавлять данные в конец файла  
 FileOutputStream fos = openFileOutput("books\_results.txt", *MODE\_APPEND*);  
 fos.write(data.getBytes());  
 fos.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void clearFile(){  
 try {  
 FileOutputStream fos = openFileOutput("books\_results.txt", *MODE\_PRIVATE*);  
 fos.close();  
 } catch (FileNotFoundException e) {  
 throw new RuntimeException(e);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
  
 private String readFromFile() {  
 StringBuilder stringBuilder = new StringBuilder();  
 try {  
 FileInputStream fis = openFileInput("books\_results.txt");  
 InputStreamReader isr = new InputStreamReader(fis, "UTF-8"); // Указываем кодировку  
 BufferedReader reader = new BufferedReader(isr);  
  
 String line;  
 while ((line = reader.readLine()) != null) {  
 stringBuilder.append(line).append("\n");  
 }  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return stringBuilder.toString();  
 }  
  
 private void logBookData(String type, String publisher, int year, int pages, String coverType) {  
 String logMessage = String.*format*(  
 "\nType: %s\tPublisher: %s\tYear: %d\tPages: %d\tCover Type: %s\n",  
 type, publisher, year, pages, coverType  
 );  
 Log.*d*("BookData", logMessage); // Логируем данные  
 Log.*d*("BookData", "------------------------------------------------------------------------------------------------"); // Логируем данные  
 }  
  
 private void logBookDataGroup(String type, String publisher, int year, int pages, String coverType, int number) {  
 String logMessage = String.*format*(  
 "\nType: %s\tPublisher: %s\tYear: %d\tPages: %d\tCover Type: %s\tNumber: %d",  
 type, publisher, year, pages, coverType, number  
 );  
 Log.*d*("BookData", logMessage); // Логируем данные  
 Log.*d*("BookData", "------------------------------------------------------------------------------------------------"); // Логируем данные  
  
 }  
  
 private void updateListView(String row) {  
  
 // Добавить новую строку  
 resultList.add(row);  
  
 // Обновить адаптер  
 adapter.notifyDataSetChanged();  
 }  
  
  
  
  
  
 @SuppressLint("Range")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 dbHelper = new DBHelper(this);  
 lvResults = findViewById(R.id.*lvResults*);  
 btnSort = findViewById(R.id.*btnSort*);  
 btnGroup = findViewById(R.id.*btnGroup*);  
 btnSum = findViewById(R.id.*btnSum*);  
 btnAvg = findViewById(R.id.*btnAvg*);  
 btnMax = findViewById(R.id.*btnMax*);  
 btnGreaterThan = findViewById(R.id.*btnGreaterThan*);  
 btnLessThanAvg = findViewById(R.id.*btnLessThanAvg*);  
 btnTypeGreaterThan = findViewById(R.id.*btnTypeGreaterThan*);  
 btnReadFromFile = findViewById(R.id.*btnReadFromFile*);  
 btnShowBooks = findViewById(R.id.*btnShowBooks*);  
  
  
 resultList = new ArrayList<>();  
 adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 SQLiteDatabase dbw = dbHelper.getWritableDatabase();  
 dbw.execSQL("DELETE FROM " + DBHelper.*TABLE\_BOOKS*);  
 dbw.execSQL("VACUUM"); // Это очищает пространство в базе данных  
  
  
 dbHelper.addBook("Учебник", "Издательство 1", 2023, 500, "Твердая");  
 dbHelper.addBook("Энциклопедия", "Издательство 1", 2022, 1500, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2024, 300, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 1", 2021, 450, "Мягкая");  
 dbHelper.addBook("Справочник", "Издательство 2", 2023, 600, "Твердая");  
 dbHelper.addBook("Словарь", "Издательство 1", 2022, 1000, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 3", 2022, 340, "Твердая");  
 dbHelper.addBook("Худ. литература", "Издательство 4", 2020, 700, "Мягкая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2021, 500, "Мягкая");  
 dbHelper.addBook("Энциклопедия", "Издательство 4", 2020, 1200, "Твердая");  
 dbHelper.addBook("Словарь", "Издательство 1", 2024, 800, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 2", 2023, 670, "Мягкая");  
 dbHelper.addBook("Справочник", "Издательство 3", 2023, 270, "Твердая");  
 dbHelper.addBook("Худ. литература", "Издательство 1", 2022, 640, "Твердая");  
 dbHelper.addBook("Учебник", "Издательство 3", 2019, 500, "Мягкая");  
  
 // Кнопка для сортировки  
// btnSort.setOnClickListener(v -> displayCursorResults(dbHelper.sortBooksByPages()));  
  
 dbw.close();  
 SQLiteDatabase dbr = dbHelper.getReadableDatabase();  
  
 btnSort.setOnClickListener(v -> {  
  
 clearFile();  
  
  
 String[] numericColumns = {"year\_of\_publication", "pages"};  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setTitle("Выберите столбец для сортировки");  
 builder.setItems(numericColumns, (dialog, which) -> {  
 String columnToSortBy = numericColumns[which];  
  
 // SQL-запрос для сортировки по выбранному столбцу  
 String query = "SELECT \* FROM books ORDER BY " + columnToSortBy;  
  
  
 resultList.clear(); // Очистка текущего списка  
 lvResults.setAdapter(adapter);  
 Cursor cursor = dbr.rawQuery(query, null);  
 if (cursor.moveToFirst()) {  
 do {  
 String type = cursor.getString(cursor.getColumnIndex("type"));  
 String publisher = cursor.getString(cursor.getColumnIndex("publisher"));  
 int year = cursor.getInt(cursor.getColumnIndex("year\_of\_publication"));  
 int pages = cursor.getInt(cursor.getColumnIndex("pages"));  
 String coverType = cursor.getString(cursor.getColumnIndex("cover\_type"));  
  
// // Формирование строки для добавления в ListView  
// String row = String.format("Type: %s, Publisher: %s, Year: %d, Pages: %d, Cover: %s",  
// type, publisher, year, pages, coverType);  
  
  
 // Логируем данные в Logcat  
 logBookData(type, publisher, year, pages, coverType);  
  
 // Записываем данные в файл  
 String dataToWrite = String.*format*("Type: %s, Publisher: %s, Year: %d, Pages: %d, Cover: %s\n",  
 type, publisher, year, pages, coverType);  
 writeToFile(dataToWrite);  
  
 } while (cursor.moveToNext());  
 updateListView("Вывод только в лог и в файл");  
 } else {  
 updateListView("No data found");  
 }  
  
 cursor.close();  
 });  
  
 builder.show();  
 });  
  
  
 // Кнопка для группировки  
// btnGroup.setOnClickListener(v -> displayCursorResults(dbHelper.groupBooksByTypeAndPublisher()));  
  
 btnGroup.setOnClickListener(v -> {  
 // Список полей для группировки  
 String[] options = {"Тип", "Издательство", "Год", "Количество страниц", "Тип обложки"};  
 boolean[] checkedItems = {false, false, false, false, false}; // Хранит информацию о выбранных элементах (по умолчанию все не выбраны)  
  
 // Создаем диалог с множественным выбором  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Выберите поля для группировки")  
 .setMultiChoiceItems(options, checkedItems, (dialog, which, isChecked) -> {  
 checkedItems[which] = isChecked; // Обновляем выбор  
 })  
 .setPositiveButton("Ок", (dialog, which) -> {  
 // Формируем запрос на основе выбранных полей  
 StringBuilder groupByColumns = new StringBuilder();  
 if (checkedItems[0]) groupByColumns.append("type, ");  
 if (checkedItems[1]) groupByColumns.append("publisher, ");  
 if (checkedItems[2]) groupByColumns.append("year\_of\_publication, ");  
 if (checkedItems[3]) groupByColumns.append("pages, ");  
 if (checkedItems[4]) groupByColumns.append("cover\_type, ");  
  
 // Убираем последнюю запятую  
 if (groupByColumns.length() > 0) {  
 groupByColumns.setLength(groupByColumns.length() - 2); // Убираем последнюю запятую и пробел  
 }  
  
 if (groupByColumns.length() == 0) {  
 // Если не выбрано ни одного поля  
 Toast.*makeText*(this, "Выберите хотя бы одно поле", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // Выполнение запроса с группировкой  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 String query = "SELECT type, publisher, year\_of\_publication, pages, cover\_type, COUNT(\*) FROM books GROUP BY " + groupByColumns;  
 Cursor cursor = db.rawQuery(query, null);  
  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 String row = "type: " + cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)) +  
 ", publisher: " + cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)) +  
 ", year: " + cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)) +  
 ", page: " + cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)) +  
 ", cover type: " + cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)) +  
 ", number: " + cursor.getInt(cursor.getColumnIndex("COUNT(\*)"));  
 resultList.add(row);  
 logBookDataGroup(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)),  
 cursor.getInt(cursor.getColumnIndex("COUNT(\*)")));  
  
 } while (cursor.moveToNext());  
  
  
 // Отображаем в логе  
  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 } else {  
 tvResults.setText("Нет данных");  
 }  
 cursor.close();  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss())  
 .show();  
 });  
  
//  
//  
//  
//  
// // Кнопка для суммы страниц  
//// btnSum.setOnClickListener(v -> displayCursorResults(dbHelper.sumPages()));  
//  
 btnSum.setOnClickListener(v -> {  
  
 clearFile();  
  
 // Список полей для выбора  
 String[] options = {"Количество страниц", "Год публикации"};  
 boolean[] checkedItems = {false, false}; // Хранит информацию о выбранных элементах (по умолчанию все не выбраны)  
  
 // Создаем диалог с множественным выбором  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Выберите поле для подсчета суммы")  
 .setSingleChoiceItems(options, -1, (dialog, which) -> {  
 // Устанавливаем выбранное поле  
 for (int i = 0; i < checkedItems.length; i++) {  
 checkedItems[i] = i == which;  
 }  
 })  
 .setPositiveButton("Ок", (dialog, which) -> {  
 String columnToSum = checkedItems[0] ? "pages" : "year\_of\_publication";  
  
 // Формируем запрос для подсчета суммы по выбранному полю  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 String query = "SELECT SUM(" + columnToSum + ") FROM books";  
 Cursor cursor = db.rawQuery(query, null);  
  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 int sum = cursor.getInt(0);  
  
 // Добавляем результат в список для отображения в ListView  
 String resultText = "Sum bu column " + columnToSum + ": " + sum;  
 resultList.add("Вывод только в файл и в лог");  
  
 // Выводим в лог  
 Log.*d*("SumResult", resultText);  
  
 // Записываем в файл  
 writeToFile(resultText);  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 } else {  
 resultList.add("Нет данных");  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
 cursor.close();  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss())  
 .show();  
 });  
  
//  
//  
// // Кнопка для среднего значения страниц  
//// btnAvg.setOnClickListener(v -> displayCursorResults(dbHelper.averagePagesByType()));  
 btnAvg.setOnClickListener(v -> {  
  
 clearFile();  
  
 // Список текстовых полей для выбора группировки (не числовые)  
 String[] options = {"type", "publisher"};  
 boolean[] checkedItems = {false, false}; // Хранит информацию о выбранных элементах (по умолчанию все не выбраны)  
  
 // Создаем диалог с множественным выбором  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Выберите поле для группировки")  
 .setMultiChoiceItems(options, checkedItems, (dialog, which, isChecked) -> {  
 checkedItems[which] = isChecked; // Обновляем выбор  
 })  
 .setPositiveButton("Ок", (dialog, which) -> {  
 // Формируем список полей для группировки  
 StringBuilder groupByColumns = new StringBuilder();  
 if (checkedItems[0]) groupByColumns.append("type, ");  
 if (checkedItems[1]) groupByColumns.append("publisher, ");  
  
 // Убираем последнюю запятую  
 if (groupByColumns.length() > 0) {  
 groupByColumns.setLength(groupByColumns.length() - 2); // Убираем последнюю запятую и пробел  
 }  
  
 if (groupByColumns.length() == 0) {  
 // Если не выбрано ни одного поля  
 Toast.*makeText*(this, "Выберите хотя бы одно поле для группировки", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 // Формируем запрос для вычисления среднего по числовым полям  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 String query = "SELECT " + groupByColumns.toString() + ", AVG(pages), AVG(year\_of\_publication) FROM books GROUP BY " + groupByColumns.toString();  
 Cursor cursor = db.rawQuery(query, null);  
  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 Log.*d*("AvgResult", "\n");  
 do {  
 // Формируем строку с результатами группировки и средними значениями  
 StringBuilder resultText = new StringBuilder();  
 for (int i = 0; i < options.length; i++) {  
 if (checkedItems[i]) {  
 resultText.append(options[i]).append(": ").append(cursor.getString(cursor.getColumnIndex(options[i].toLowerCase()))).append(", ");  
 }  
 }  
 resultText.append("Average number of pages: ").append(cursor.getInt(cursor.getColumnIndex("AVG(pages)")))  
 .append(", Average year of publication: ").append(cursor.getInt(cursor.getColumnIndex("AVG(year\_of\_publication)")));  
  
 resultList.add(resultText.toString());  
  
 // Выводим в лог  
 Log.*d*("AvgResult", resultText.toString());  
 Log.*d*("AvgResult", "\n");  
  
 // Записываем в файл  
 writeToFile(resultText.toString());  
 writeToFile("\n");  
 } while (cursor.moveToNext());  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 } else {  
 resultList.add("Нет данных");  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
 cursor.close();  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss())  
 .show();  
 });  
  
  
  
  
  
//  
//  
// // Кнопка для максимального значения страниц  
//// btnMax.setOnClickListener(v -> displayCursorResults(dbHelper.maxPages()));  
//  
 btnMax.setOnClickListener(v -> {  
 // Создаем список для хранения результатов  
 ArrayList<String> resultList = new ArrayList<>();  
  
 // Запрашиваем максимальные значения для страниц и года публикации  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursorPages = db.rawQuery("SELECT MAX(pages) FROM books", null);  
 Cursor cursorYear = db.rawQuery("SELECT MAX(year\_of\_publication) FROM books", null);  
  
 // Если курсоры не пустые и содержат данные  
 if (cursorPages != null && cursorPages.moveToFirst() && cursorYear != null && cursorYear.moveToFirst()) {  
 int maxPages = cursorPages.getInt(0);  
 int maxYear = cursorYear.getInt(0);  
  
 // Запрос для поиска записей, где максимальные значения  
 String query = "SELECT \* FROM books WHERE pages = ? OR year\_of\_publication = ?";  
 Cursor cursorResults = db.rawQuery(query, new String[]{String.*valueOf*(maxPages), String.*valueOf*(maxYear)});  
  
 if (cursorResults != null && cursorResults.moveToFirst()) {  
 do {  
 StringBuilder resultText = new StringBuilder();  
  
 // Сравниваем максимальные значения и выводим соответствующие данные  
 if (maxPages == maxYear) {  
 resultText.append("Max pages and year: ")  
 .append("pages: ").append(maxPages)  
 .append(", year: ").append(maxYear)  
 .append("\n");  
 } else {  
 if (cursorResults.getInt(cursorResults.getColumnIndex(DBHelper.*COLUMN\_PAGES*)) == maxPages) {  
 resultText.append("Макс. страницы: ").append(maxPages).append("\n");  
 }  
 if (cursorResults.getInt(cursorResults.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)) == maxYear) {  
 resultText.append("Макс. год: ").append(maxYear).append("\n");  
 }  
 }  
  
 // Формируем строку с данными записи  
 resultText.append("type: ").append(cursorResults.getString(cursorResults.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", publisher: ").append(cursorResults.getString(cursorResults.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", year: ").append(cursorResults.getInt(cursorResults.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", pages: ").append(cursorResults.getInt(cursorResults.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", cover type: ").append(cursorResults.getString(cursorResults.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
  
 // Отображаем в логе  
 Log.*d*("MaxValues", "\n");  
 Log.*d*("MaxValues", resultText.toString());  
 Log.*d*("MaxValues", "------------------------------------------------------------------------------");  
  
 } while (cursorResults.moveToNext());  
  
 resultList.clear();  
  
 resultList.add("Вывод только в лог");  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
  
 cursorResults.close();  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
  
 cursorPages.close();  
 cursorYear.close();  
 });  
  
//  
//  
//  
//  
//  
// // Кнопка для количества страниц меньше средней  
//// btnLessThanAvg.setOnClickListener(v -> displayCursorResults(dbHelper.booksWithPagesLessThanAverage()));  
//  
 btnLessThanAvg.setOnClickListener(v -> {  
 // Получаем среднее значение для страниц  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 Cursor cursorAvgPages = db.rawQuery("SELECT AVG(pages) FROM books", null);  
 Cursor cursorAvgYear = db.rawQuery("SELECT AVG(year\_of\_publication) FROM books", null);  
  
 float avgPages = 0;  
 float avgYear = 0;  
  
 // Получаем среднее значение для pages, если есть данные  
 if (cursorAvgPages != null && cursorAvgPages.moveToFirst()) {  
 avgPages = cursorAvgPages.getFloat(0);  
 }  
  
 // Получаем среднее значение для year\_of\_publication, если есть данные  
 if (cursorAvgYear != null && cursorAvgYear.moveToFirst()) {  
 avgYear = cursorAvgYear.getFloat(0);  
 }  
  
 // Выполняем запрос для получения всех записей, где числовые поля меньше среднего  
 Cursor cursor = db.rawQuery("SELECT \* FROM books WHERE pages < ? AND year\_of\_publication < ?",  
 new String[]{String.*valueOf*(avgPages), String.*valueOf*(avgYear)});  
  
 // Список для отображения результатов  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 StringBuilder resultText = new StringBuilder();  
 resultText.append("type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", publisher: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", year: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", pages: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", cover type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
 resultList.add(resultText.toString());  
  
 logBookData(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)));  
  
 } while (cursor.moveToNext());  
  
 Log.*d*("bookData", "\n");  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
  
 cursor.close();  
 cursorAvgPages.close();  
 cursorAvgYear.close();  
 });  
  
//  
//  
// // Кнопка для типа книг с страницами больше 300  
//// btnTypeGreaterThan.setOnClickListener(v -> displayCursorResults(dbHelper.bookTypesWithPagesGreaterThan(300)));  
//  
 btnTypeGreaterThan.setOnClickListener(v -> {  
 // Создаем контейнер для EditText и Spinner  
 LinearLayout layout = new LinearLayout(this);  
 layout.setOrientation(LinearLayout.*VERTICAL*);  
  
 // Создаем EditText для ввода числа  
 final EditText inputValue = new EditText(this);  
 inputValue.setInputType(InputType.*TYPE\_CLASS\_NUMBER*); // Устанавливаем тип ввода как число  
 inputValue.setHint("Введите число");  
  
 // Добавляем EditText в контейнер  
 layout.addView(inputValue);  
  
 // Создаем диалоговое окно  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Введите число и выберите поля")  
 .setView(layout)  
 .setPositiveButton("Ок", (dialog, which) -> {  
 // Получаем введенное число  
 String inputText = inputValue.getText().toString();  
 if (!inputText.isEmpty()) {  
 int value = Integer.*parseInt*(inputText); // Преобразуем строку в число  
  
 // Запрос для получения записей, где оба числовых поля больше введенного числа  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 String query = "SELECT \* FROM books WHERE pages > ? AND year\_of\_publication > ?";  
 Cursor cursor = db.rawQuery(query, new String[]{String.*valueOf*(value), String.*valueOf*(value)});  
  
 // Список для отображения результатов  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 StringBuilder resultText = new StringBuilder();  
 resultText.append("type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", publisher: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", year: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", pages: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", cover type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
 resultList.add(resultText.toString());  
  
 logBookData(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)),  
 cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)),  
 cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)));  
  
 } while (cursor.moveToNext());  
  
 Log.*d*("bookData", "\n");  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
 cursor.close();  
 } else {  
 // Если поле пустое  
 Toast.*makeText*(this, "Пожалуйста, введите число", Toast.*LENGTH\_SHORT*).show();  
 }  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss())  
 .show();  
 });  
  
  
  
 btnGreaterThan.setOnClickListener(v -> {  
 // Создаем контейнер для EditText и Spinner  
 LinearLayout layout = new LinearLayout(this);  
 layout.setOrientation(LinearLayout.*VERTICAL*);  
  
 // Создаем EditText для ввода числа  
 final EditText inputValue = new EditText(this);  
 inputValue.setInputType(InputType.*TYPE\_CLASS\_NUMBER*); // Устанавливаем тип ввода как число  
 inputValue.setHint("Введите число");  
  
 // Создаем Spinner для выбора поля для отображения  
 final Spinner fieldSpinner = new Spinner(this);  
 ArrayAdapter<CharSequence> adapterSpinner = ArrayAdapter.*createFromResource*(this,  
 R.array.*fields\_array*, android.R.layout.*simple\_spinner\_item*);  
 adapterSpinner.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 fieldSpinner.setAdapter(adapterSpinner);  
  
 // Добавляем EditText и Spinner в контейнер  
 layout.addView(inputValue);  
 layout.addView(fieldSpinner);  
  
 // Создаем диалоговое окно  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Введите число и выберите поле для отображения")  
 .setView(layout)  
 .setPositiveButton("Ок", (dialog, which) -> {  
 // Получаем введенное число  
 String inputText = inputValue.getText().toString();  
 if (!inputText.isEmpty()) {  
 int value = Integer.*parseInt*(inputText); // Преобразуем строку в число  
  
 // Получаем выбранное поле для отображения  
 String selectedField = fieldSpinner.getSelectedItem().toString();  
  
 // Запрос для получения записей, где хотя бы одно числовое поле больше введенного числа  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
 String query = "SELECT \* FROM books WHERE pages > ? OR year\_of\_publication > ?";  
 Cursor cursor = db.rawQuery(query, new String[]{String.*valueOf*(value), String.*valueOf*(value)});  
  
 // Список для отображения результатов  
 ArrayList<String> resultList = new ArrayList<>();  
 if (cursor != null && cursor.moveToFirst()) {  
 Log.*d*("GreaterThanPages", "\n");  
 do {  
 StringBuilder resultText = new StringBuilder();  
 if (selectedField.equals("type")) {  
 resultText.append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)));  
 } else if (selectedField.equals("publisher")) {  
 resultText.append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)));  
 } else if (selectedField.equals("year\_of\_publication")) {  
 resultText.append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)));  
 } else if (selectedField.equals("pages")) {  
 resultText.append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)));  
 } else if (selectedField.equals("cover\_type")) {  
 resultText.append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)));  
 }  
  
  
  
 // Отображаем в логе  
 Log.*d*("GreaterThanPages", resultText.toString());  
 Log.*d*("GreaterThanPages", "----------------------------");  
  
 } while (cursor.moveToNext());  
  
 resultList.clear();  
  
 resultList.add("Вывод только в лог");  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
 cursor.close();  
 } else {  
 // Если поле пустое  
 Toast.*makeText*(this, "Пожалуйста, введите число", Toast.*LENGTH\_SHORT*).show();  
 }  
 })  
 .setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss())  
 .show();  
 });  
  
  
  
  
  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
 btnReadFromFile.setOnClickListener(v -> {  
 // Читаем данные из файла  
 String fileData = readFromFile();  
  
 // Разделяем данные по строкам  
 String[] books = fileData.split("\n");  
  
 // Создаем список, чтобы поместить данные в ListView  
 List<String> bookList = new ArrayList<>(Arrays.*asList*(books));  
  
 // Создаем ArrayAdapter для отображения данных в ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(MainActivity.this, android.R.layout.*simple\_list\_item\_1*, bookList);  
  
 // Устанавливаем адаптер в ListView  
 lvResults.setAdapter(adapter);  
 });  
  
  
 btnShowBooks.setOnClickListener(v -> {  
 // Открываем базу данных для чтения  
 SQLiteDatabase db = dbHelper.getReadableDatabase();  
  
 // Выполняем запрос для получения всех записей  
 Cursor cursor = db.rawQuery("SELECT \* FROM books", null);  
  
 // Список для хранения строк с результатами  
 ArrayList<String> resultList = new ArrayList<>();  
  
 // Проверяем, есть ли данные в таблице  
 if (cursor != null && cursor.moveToFirst()) {  
 do {  
 // Создаем строку для каждой записи  
 StringBuilder resultText = new StringBuilder();  
 resultText.append("type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_TYPE*)))  
 .append(", publisher: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_PUBLISHER*)))  
 .append(", year: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*)))  
 .append(", pages: ").append(cursor.getInt(cursor.getColumnIndex(DBHelper.*COLUMN\_PAGES*)))  
 .append(", cover type: ").append(cursor.getString(cursor.getColumnIndex(DBHelper.*COLUMN\_COVER\_TYPE*)))  
 .append("\n");  
  
 // Добавляем строку в список  
 resultList.add(resultText.toString());  
 } while (cursor.moveToNext());  
  
 // Обновляем ListView  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
  
 // Логируем все результаты  
 Log.*d*("AllBooks", resultList.toString());  
 } else {  
 resultList.add("Нет данных");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_list\_item\_1*, resultList);  
 lvResults.setAdapter(adapter);  
 }  
  
 // Закрываем курсор  
 cursor.close();  
 });

**DBHelper.java**

package com.example.android\_dev\_lab4new;  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
public class DBHelper extends SQLiteOpenHelper {  
  
 private static final String *DATABASE\_NAME* = "books.db";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 public static final String *TABLE\_BOOKS* = "books";  
 public static final String *COLUMN\_ID* = "id";  
 public static final String *COLUMN\_TYPE* = "type";  
 public static final String *COLUMN\_PUBLISHER* = "publisher";  
 public static final String *COLUMN\_YEAR\_OF\_PUBLICATION* = "year\_of\_publication";  
 public static final String *COLUMN\_PAGES* = "pages";  
 public static final String *COLUMN\_COVER\_TYPE* = "cover\_type";  
  
 public DBHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String CREATE\_TABLE = "CREATE TABLE " + *TABLE\_BOOKS* + " ("  
 + *COLUMN\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT, "  
 + *COLUMN\_TYPE* + " TEXT, "  
 + *COLUMN\_PUBLISHER* + " TEXT, "  
 + *COLUMN\_YEAR\_OF\_PUBLICATION* + " INTEGER, "  
 + *COLUMN\_PAGES* + " INTEGER, "  
 + *COLUMN\_COVER\_TYPE* + " TEXT)";  
 db.execSQL(CREATE\_TABLE);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_BOOKS*);  
 onCreate(db);  
 }  
  
// public Cursor sortBooksByPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " ORDER BY " + DBHelper.COLUMN\_PAGES, null);  
// }  
//  
// public Cursor groupBooksByTypeAndPublisher() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT type, publisher, COUNT(\*) FROM " + DBHelper.TABLE\_BOOKS + " GROUP BY type, publisher", null);  
// }  
// public Cursor sumPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT SUM(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS, null);  
// }  
//  
// public Cursor averagePagesByType() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT type, AVG(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + " GROUP BY type", null);  
// }  
//  
// public Cursor maxPages() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " = (SELECT MAX(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + ")", null);  
// }  
// public Cursor booksWithPagesGreaterThan(int threshold) {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " > ?", new String[]{String.valueOf(threshold)});  
// }  
//  
//  
// public Cursor booksWithPagesLessThanAverage() {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT \* FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " < (SELECT AVG(" + DBHelper.COLUMN\_PAGES + ") FROM " + DBHelper.TABLE\_BOOKS + ")", null);  
// }  
// public Cursor bookTypesWithPagesGreaterThan(int threshold) {  
// SQLiteDatabase db = getReadableDatabase();  
// return db.rawQuery("SELECT " + DBHelper.COLUMN\_TYPE + " FROM " + DBHelper.TABLE\_BOOKS + " WHERE " + DBHelper.COLUMN\_PAGES + " > ?", new String[]{String.valueOf(threshold)});  
// }  
  
 public void addBook(String type, String publisher, int year, int pages, String coverType) {  
 SQLiteDatabase db = getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put(DBHelper.*COLUMN\_TYPE*, type);  
 values.put(DBHelper.*COLUMN\_PUBLISHER*, publisher);  
 values.put(DBHelper.*COLUMN\_YEAR\_OF\_PUBLICATION*, year);  
 values.put(DBHelper.*COLUMN\_PAGES*, pages);  
 values.put(DBHelper.*COLUMN\_COVER\_TYPE*, coverType);  
 db.insert(DBHelper.*TABLE\_BOOKS*, null, values);  
 db.close();  
 }  
  
 public Cursor getAllBooks() {  
 SQLiteDatabase db = getReadableDatabase();  
 return db.rawQuery("SELECT \* FROM " + DBHelper.*TABLE\_BOOKS*, null);  
 }  
  
  
}

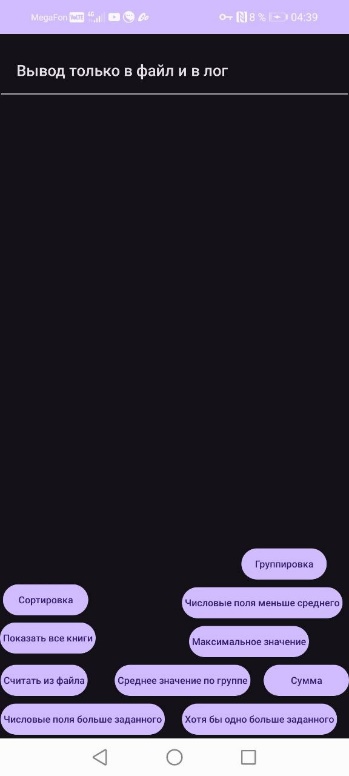
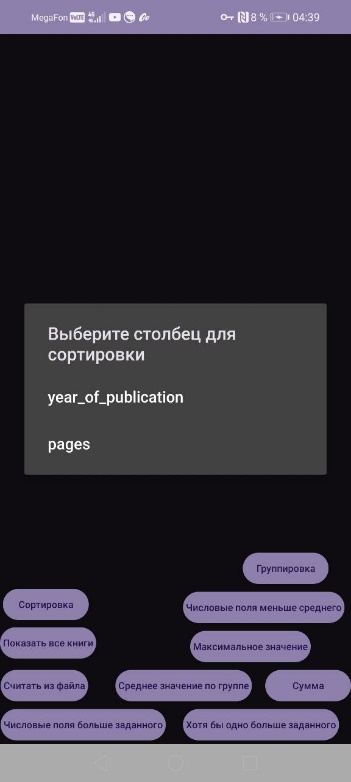
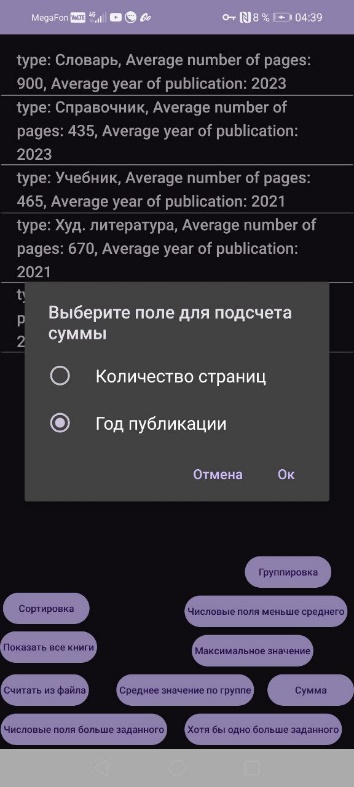
**activity\_main.xml**

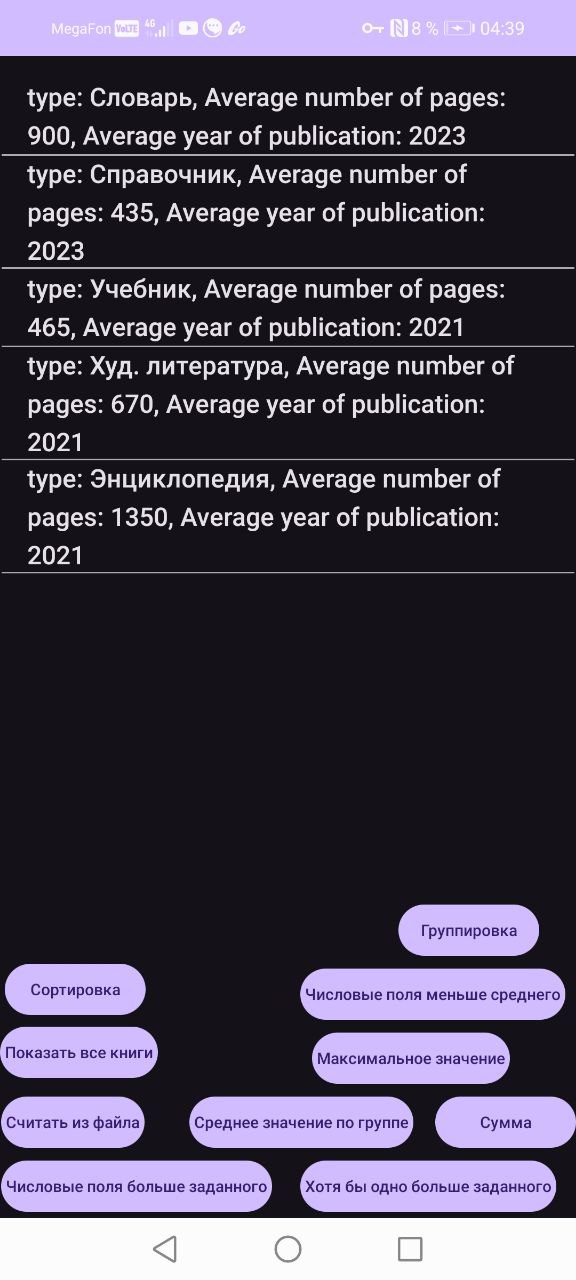
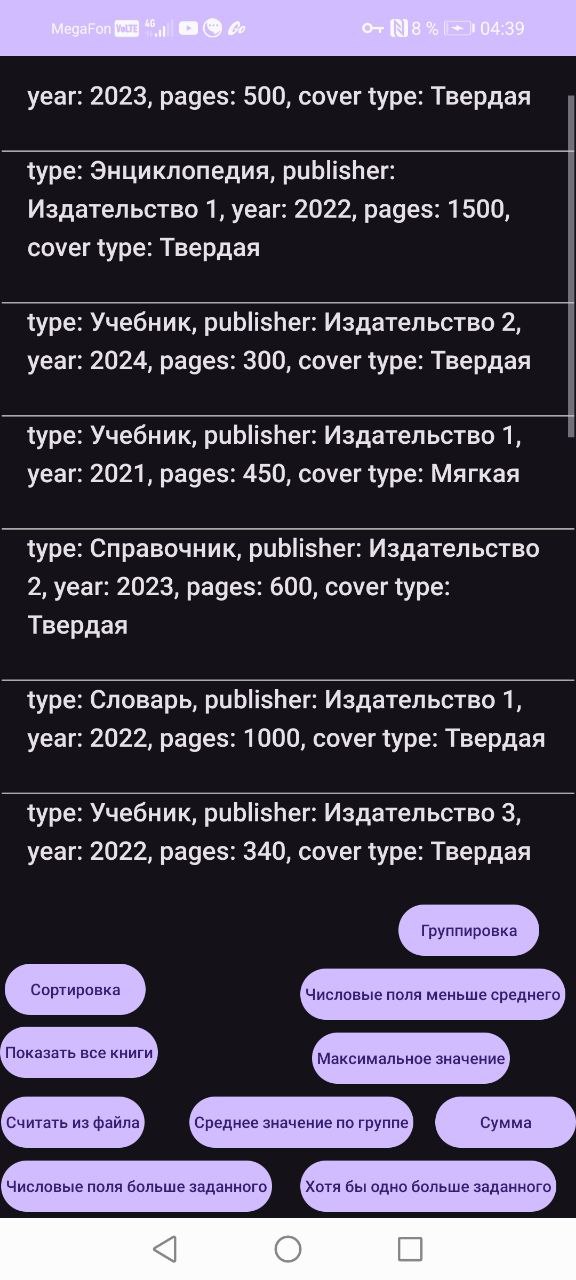
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <!-- Кнопки для запросов -->  
  
 <!-- Кнопки для добавления книги и вывода всех книг -->  
  
 <!-- TextView для отображения результатов -->  
  
 <Button  
 android:id="@+id/btnSort"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Сортировка по страницам"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.011"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.821" />  
  
 <Button  
 android:id="@+id/btnShowBooks"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Показать все книги"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnReadFromFile"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.994" />  
  
 <Button  
 android:id="@+id/btnLessThanAvg"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Количество страниц меньше Среднего"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnMax"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.966"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnSum"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Сумма страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnReadFromFile"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Считать из файла"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnTypeGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.003"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnMax"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Максимальное количество страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnAvg"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.825"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnTypeGreaterThan"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Тип книг где больше 300 страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.004"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnGreaterThan"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Количество страниц больше 300"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.939"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnAvg"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Среднее количество страниц"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnGreaterThan"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.538"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btnGroup"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="40dp"  
 android:padding="3dp"  
 android:text="Группировка по типу и издательству"  
 android:textSize="10sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnLessThanAvg"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.916"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <TextView  
 android:id="@+id/tvResults"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Результаты будут здесь"  
 android:textSize="12sp"  
 app:layout\_constraintBottom\_toTopOf="@id/btnShowBooks"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Android\_dev\_lab4new"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Результаты выполнения работы:**

**  **

** **

**Вывод:** в ходе лабораторной работы было разработано приложение, взаимодействующее с базой данных SQLite.