



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікського»

Лабораторна робота № 2

З дисципліни «Бази даних і засоби управління»

**Назва: «Створення додатку бази даних, орієнтованого на взаємодію з
СУБД PostgreSQL»**

Виконав студент групи: KB-84
ПІБ: Мельник Ярослав Володимирович

Київ 2020

Технічне завдання

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

URL репозиторію: <https://github.com/YaroslavMelnyk/Database/tree/lab2>

Завдання 1:

Додавання даних:

```
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 8
Input table: reader
2          Джейсон          Стетхем          js@js.com
3          Емілія           Кларк            ek@ek.com
4          Скарлет          Йохансон         sj@sg.com
5          Джоні            Ден              jd@jd.com
6          Кіану            Півз             kr@kr.com
9          qwerty           asdfggh          zxcvbn
11         qwerty           asdfgh           zxcvbn
474        NCCWLN           UKBRYA           OMYSAX
937        PFPATA           HEQSWR           ODOORA
482        CMLEJK           KIBOGH           KJJUTX
142        EAQVNB           BYLPSL           QQNUJL
756        OQPGHI           NKWFAB           NILHSX
889        ASCQSI           TWJJBX           FRBSCN
952        VBXTWO           RGVJIO           WRFNPQ
111        WSOEAI           WFPKJ           FEHRIO
786        EBKDXV           HORMXV           DTTNMB
698        XNMWOO           PJTQAL           QFGUHI
1007       BHYKOU           UEKVVL           KTSORP
799        YDPYCL           LJOESG           KMKFRC
908        FALFHU           KADVPJ           CLVWHN
68         ILTJGW           LNNROS           MSMXWJ
210        EHYOXF           THYSKA           MUQKBH
136        FLGNKI           WUFJQV           NODYBR
399        GUDCFA           BBRCJG           BFLJQX
130        FRAIQQ           NCRMEQ           AIDHMK
567        NYXERF           CMJUUT          LJUWAL
602        GAMXSJ           MBTQYA           EYVKCN
```

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 1
Input table: reader
Input reader_id: 10
Input reader_name: qazwsx
Input reader_surname: edcrfv
Input reader_email: tgbyhn

Operation is performed
```

```
9. View table columns name
10. Exit
Enter the menu number: 8
Input table: reader
2          Джейсон          Стетхем          js@js.com
3          Емілія          Кларк          ek@ek.com
4          Скарлет          Йохансон        sj@sg.com
5          Джоні          Ден          jd@jd.com
6          Кіану          Піз          kr@kr.com
9          qwerty          asdfggh        zxcvbn
10         qazwsx          edcrfv         tgbyhn
11         qwerty          asdfgh         zxcvbn
474        NCCMLN          UKBRYA         QMYSAX
937        PFPATA          HEQSHR         ODOORA
482        CMLEJK          KIBOGH         KJJUTX
142        EAQVNB          BYLPSL         QQNUJL
756        OQPGHI          NKWFAB         NILHSX
889        ASCQSI          TMUJXB         FRBSCN
952        VBXTWO          RGYJIO         WRFNPQ
111        WSOEAI          WFPKJ          FEHRIO
786        EBKDXV          HORPKV         DTTNMB
698        XNMWOO          PJTQAL         QFGUHI
1007       BHYKOU          UEKKVL         KTSORP
799        YDPYCL          LJOESG         KMKFRC
908        FALFHU          KADYPJ         CLVvHN
68         ILTJGW          LNNROS         MSNMXX
210        EHYOXF          THYSKA         MUQKBH
136        FLGNKI          WUFJQV         NODYBR
399        GUDCFA          BBRJIG         BFLJQX
130        FRATQQ          NCRMEQ         AIDHwK
567        NYXERF          CMJUJJ         LDUWAL
602        GAMXSJ          MBTQYA         EYVKCN
```

Обробка помилок:

Введення даних з уже існуючим первинним ключем

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 1
Input table: reader
Input reader_id: 10
Input reader_name: qazwsx
Input reader_surname: edcrfv
Input reader_email: tgbyhn

Error: Incorrect data entered

Operation is not performed
```

Введення даних, що не існує в батьківській таблиці:

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 1
Input table: reader_ticket
Input reader_id: 12
Input book_id: 10
Input date_issued: 2020-10-20
Input validity: 30
|
Error: Incorrect data entered

Operation is not performed
```

Видалення даних:

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 2
Input table: reader
Input the name of the column by which the data will be displayed: reader_id
Input value column: 10
|
Operation is performed

```

```

6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 8
Input table: reader

```

| | | | |
|------|---------|----------|-----------|
| 2 | Джейсон | Стетхем | js@js.com |
| 3 | Емілія | Кларк | ek@ek.com |
| 4 | Скарлет | Йохансон | sj@sg.com |
| 5 | Джоні | Деп | jd@jd.com |
| 6 | Кіану | Ріоз | kr@kr.com |
| 9 | qwerty | asdfggh | zxcvbn |
| 11 | qwerty | asdfgh | zxcvbn |
| 474 | NCCWLN | UKBRYA | OMYSAX |
| 937 | PPFATA | HEQSWR | ODOORA |
| 482 | CMLEJK | KIBOGH | KJJJTX |
| 142 | EAQVHB | BYLPSL | QQNUJL |
| 756 | OQPGHI | NKWFAB | NILHSX |
| 889 | ASCQSI | TWUJXB | FRBSCN |
| 952 | VBXTWO | RGYJIO | WRFNPQ |
| 111 | WSOEAI | WFPKJ | FEHRIO |
| 786 | EBKDXV | HORMXV | DTTNMB |
| 698 | XNMWOO | PJTQAL | QFGUHI |
| 1007 | BHYKOU | UEKKVL | KTSORP |
| 799 | YDPYCL | LJOESG | KMKFRC |
| 908 | FALFHU | KADVPJ | CLVWHN |
| 68 | ILTJGW | LNNROS | MSMIXW |
| 210 | EHYOXC | THYSKA | MUQKBH |
| 136 | FLGNKI | WUFJQV | NODYBR |
| 399 | GUDCFA | BBRCJG | BFLJQX |
| 130 | FRAIQQ | NCRMEQ | AIDHMK |
| 567 | NYXERF | CMJUUD | LJUMAL |
| 602 | GAHXSJ | MBTQYA | EYWKCN |

Обробка помилок:

Видалення даних з таблиці reader, при цьому у підлеглий таблиці reader_ticket є зв'язані дані:

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 2
Input table: reader
Input the name of the column by which the data will be displayed: reader_id
Input value column: 68
|
Error: Incorrect data entered

Operation is not performed

```

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 8
Input table: reader_ticket

```

| | | | |
|------|-----|------------|----|
| 952 | 69 | 2012-12-26 | 16 |
| 799 | 29 | 2013-08-24 | 16 |
| 136 | 251 | 2016-02-02 | 19 |
| 4 | 6 | 2017-07-26 | 21 |
| 68 | 359 | 2017-08-21 | 21 |
| 5 | 4 | 2018-01-15 | 22 |
| 908 | 198 | 2018-12-26 | 23 |
| 1007 | 42 | 2020-07-18 | 25 |
| 952 | 229 | 2021-10-18 | 26 |
| 3 | 263 | 2022-08-18 | 27 |
| 908 | 171 | 2022-08-23 | 27 |
| 6 | 54 | 2026-12-03 | 32 |
| 698 | 4 | 2027-07-20 | 33 |
| 952 | 6 | 2028-04-22 | 34 |
| 952 | 205 | 2029-06-30 | 35 |
| 4 | 4 | 2031-05-28 | 38 |
| 698 | 268 | 2032-01-13 | 38 |
| 111 | 132 | 2032-03-09 | 39 |
| 210 | 23 | 2036-07-09 | 44 |
| 786 | 42 | 2040-03-13 | 48 |
| 482 | 7 | 2013-02-13 | 16 |
| 786 | 4 | 2013-02-23 | 16 |
| 602 | 132 | 2013-04-04 | 16 |
| 9 | 42 | 2013-05-02 | 16 |
| 4 | 23 | 2014-11-05 | 18 |

Модифікація даних:

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 3
Input table: reader
Input the name of the column in which the data changes: reader_id
Input value column: 11
Input reader_id: 10
Input reader_name: uiop
Input reader_surname: jlk
Input reader_email: m

Operation is performed

```

```

4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 8
Input table: reader

```

| | | | |
|------|---------|----------|-----------|
| 2 | Джейсон | Стетхем | js@js.com |
| 3 | Емілія | Кларк | ek@ek.com |
| 4 | Скарлет | Йохансон | sj@sg.com |
| 5 | Джоні | Деп | jd@jd.com |
| 6 | Кіану | Пізз | kr@kr.com |
| 9 | qwerty | asdfggh | zxcvbbn |
| 474 | NCCWLN | UKBRYA | OMYSAX |
| 937 | PFPATA | HEQSWR | ODOORA |
| 482 | CMLEJK | KTBQGH | KJJUTX |
| 142 | EAQVHB | BYLPSL | QQNUJL |
| 756 | OQPGHI | NKWFAB | NILHSX |
| 889 | ASCQSI | TWUJXB | FRBSCN |
| 952 | VBXTWO | RGVJIO | WRFNPQ |
| 111 | WSOEAI | WFPKJ | FEHRTIO |
| 786 | EBKDXV | HORMXV | DTTNMB |
| 698 | XNMWOO | PJTQAL | QFGUHI |
| 1007 | BHYKOU | UEKKVL | KTSORP |
| 799 | YDPYCL | LJOESG | KMKFRC |
| 908 | FALFHU | KADYPJ | CLVWHN |
| 68 | ILTJGW | LNNROS | MSNMXW |
| 210 | EHYXOF | THYSKA | MUQKBH |
| 136 | FLGNKI | WUFJQV | NODYBR |
| 399 | GUDCFA | BBRCJG | BFLJQX |
| 130 | FRAIQQ | NCRMEQ | AIDHMK |
| 567 | NYXERF | CMJUJ | LJUWAL |
| 602 | GAMXSJ | MBTQYA | EYVKCN |
| 10 | uiop | jlk | m |

Обробка помилок:

Перетворення даних на вже існуючі

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 3
Input table: reader
Input the name of the column in which the data changes: reader_id
Input value column: 10
Input reader_id: 9
Input reader_name: qwerty
Input reader_surname: asdfgh
Input reader_email: zxcvbn
|
Error: Incorrect data entered

Operation is not performed

```

Завдання 2:

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 4
Input table( all tables, input 'all'): reader_ticket
Input the amount of random data: 100000

Operation is performed
```

| | | | |
|-------|-------|------------|------|
| 38975 | 88 | 2738-06-07 | 8680 |
| 889 | 11719 | 2738-06-15 | 8680 |
| 510 | 54 | 2738-06-17 | 8680 |
| 7804 | 5648 | 2738-06-20 | 8680 |
| 833 | 58179 | 2738-06-21 | 8680 |
| 3221 | 42 | 2738-06-26 | 8680 |
| 9287 | 3653 | 2738-06-26 | 8680 |
| 89589 | 24567 | 2738-06-29 | 8680 |
| 7256 | 1500 | 2738-06-30 | 8680 |
| 4450 | 23 | 2738-07-05 | 8680 |
| 9124 | 38501 | 2738-07-07 | 8681 |
| 2410 | 6680 | 2738-07-10 | 8681 |
| 5 | 6069 | 2738-07-12 | 8681 |
| 44571 | 54 | 2738-07-15 | 8681 |
| 9329 | 82 | 2738-07-18 | 8681 |
| 67345 | 3005 | 2738-07-24 | 8681 |
| 9415 | 219 | 2738-07-26 | 8681 |
| 4552 | 46061 | 2738-08-01 | 8681 |
| 11119 | 172 | 2738-08-03 | 8681 |
| 3487 | 23160 | 2738-08-03 | 8681 |
| 29882 | 82 | 2738-08-09 | 8682 |
| 5394 | 58179 | 2738-08-10 | 8682 |
| 46260 | 7273 | 2738-08-11 | 8682 |
| 738 | 18787 | 2738-08-13 | 8682 |
| 889 | 63303 | 2738-08-17 | 8682 |
| 11347 | 67310 | 2738-08-22 | 8682 |
| 6559 | 63303 | 2738-08-24 | 8682 |

1

SELECT count(*) FROM reader_ticket

2

3

| Результат | План выполн... | Сообщения | Notific |
|--|----------------|-----------|---------|
| <div><div></div><div>count</div><div>bigint</div><div></div></div> | | | |
| 1 | 100000 | | |


```

1 INSERT INTO book (book_id, book_name, author_id, year_publication, count_books)
2 (SELECT trunc(random()*10000)::int,
3 chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
4 chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
5 chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),
6 author_id,
7 trunc(1960 + random()*150)::int,
8 trunc(random()*10000)::int
9 FROM author tablesample BERNOULLI(100)
10 ORDER BY random()) LIMIT 10

```

Результат План выполн... Сообщения Notifications

| | reader_id [PK] integer | book_id [PK] integer | date_issued date | validity integer |
|---|---------------------------|-------------------------|---------------------|---------------------|
| 1 | 5762 | 8397 | 2012-01-01 | 15 |
| 2 | 52673 | 357 | 2012-01-04 | 15 |
| 3 | 597 | 18033 | 2012-01-12 | 15 |
| 4 | 28551 | 67739 | 2012-01-15 | 15 |
| 5 | 889 | 3005 | 2012-01-16 | 15 |
| 6 | 11871 | 101 | 2012-01-17 | 15 |
| 7 | 99321 | 41921 | 2012-01-17 | 15 |
| 8 | 63667 | 9892 | 2012-01-24 | 15 |

```

randomReader = "INSERT INTO reader (reader_id, reader_name, reader_surname, reader_email)"
+ "SELECT trunc(10 + random()*10000000)::int, "
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
+ "FROM generate_series(1, " + numberColumns + ")";

```

```

randomAuthor = "INSERT INTO author (author_id, author_name, author_surname, author_email)"
+ "SELECT trunc(10 + random()*10000)::int, "
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
+ "FROM generate_series(1, " + numberColumns + ")";

```

```

randomBook = "INSERT INTO book (book_id, book_name, author_id, year_publication, count_books)\r\n"
+ "(SELECT trunc(random()*10000)::int,\r\n"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||\r\n"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||\r\n"
+ "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),\r\n"
+ "author_id,\r\n"
+ "trunc(1960 + random()*150)::int,\r\n"
+ "trunc(random()*10000)::int\r\n"
+ "FROM author tablesample BERNOULLI(100)\r\n"
+ "ORDER BY random()) LIMIT " + numberColumns;

```

```
CREATE TABLE randomReaderTicket (INT numberColumns) {
randomReaderTicket = "INSERT INTO reader_ticket (reader_id, book_id, date_issued, validity)\r\n"
+ "(SELECT reader_id, book_id,\r\n"
+ "date '2012-01-01' + trunc(random() * 100 * (date '2020-05-20' - date '2012-01-01'))::int,\r\n"
+ "trunc(15 + random()*10000)::int\r\n"
+ "FROM reader, book tablesample BERNOULLI(100))\r\n"
+ "ORDER BY random()) LIMIT " + numberColumns;
```

Завдання 3

Пошук книги за ім'ям автора

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 5
Input name: Джоан
Джоан          Роулінг          Гаррі Поттер І в'язень Азкабану
Джоан          Роулінг          Гаррі Поттер і філософський камінь
Джоан          Роулінг          CCCCCC
Джоан          Роулінг          CCCCCC
Джоан          Роулінг          BBBBBB
Джоан          Роулінг          EEEEE
Джоан          Роулінг          BBBBBB
Джоан          Роулінг          TTTTTT
Джоан          Роулінг          RRRRRR
Execution Time: 0.083 ms
```

```
1 SELECT a.author_name, a.author_surname, book_name
2 FROM book AS b INNER JOIN
3 (SELECT author_id, author_name, author_surname
4 FROM author WHERE author_name LIKE 'Джоан')
5 AS a ON b.author_id = a.author_id
```

| | Результат | План выполн... | Сообщения | Notifications |
|---|---------------------|------------------------|------------------------|---------------|
| | author_name name | author_surname name | book_name text | |
| 1 | Джоан | Роулінг | Гаррі Поттер І в'яз... | |
| 2 | Джоан | Роулінг | Гаррі Поттер і філо... | |
| 3 | Джоан | Роулінг | CCCCCC | |
| 4 | Джоан | Роулінг | CCCCCC | |
| 5 | Джоан | Роулінг | BBBBBB | |
| 6 | Джоан | Роулінг | EEEEEE | |
| 7 | Джоан | Роулінг | BBBBBB | |
| 8 | Джоан | Роулінг | TTTTTT | |
| 9 | Джоан | Роулінг | RRRRRR | |

```

1  EXPLAIN ANALYZE
2  SELECT a.author_name, a.author_surname, book_name
3  FROM book AS b INNER JOIN
4  (SELECT author_id, author_name, author_surname
5  FROM author WHERE author_name LIKE 'Джоан')
6  AS a ON b.author_id = a.author_id

```

Результат План выполн... Сообщения Notifications

| | QUERY PLAN text | |
|----|---|--|
| 1 | Hash Join (cost=4.70..8.70 rows=3 width=136) (actual time=0.047..0.066 rows=9 loops=1) | |
| 2 | Hash Cond: (b.author_id = author.author_id) | |
| 3 | -> Seq Scan on book b (cost=0.00..3.56 rows=156 width=12) (actual time=0.009..0.017 rows=156 loops=1) | |
| 4 | -> Hash (cost=4.69..4.69 rows=1 width=132) (actual time=0.032..0.032 rows=1 loops=1) | |
| 5 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 6 | -> Seq Scan on author (cost=0.00..4.69 rows=1 width=132) (actual time=0.023..0.030 rows=1 loops=1) | |
| 7 | Filter: (author_name ~~ 'Джоан':text) | |
| 8 | Rows Removed by Filter: 54 | |
| 9 | Planning Time: 0.170 ms | |
| 10 | Execution Time: 0.080 ms | |

Пошук читачів за кількістю взятих книг за певний період

```

1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 6
Input date(YYYY-MM-DD): 2015-10-10
UHLPGV                LGWITA                2
HKDDPI                LEATXI                2
NCNYCB                SPONHJ                1
EBKDXV                HORMXV                1
UQPMMR                CBATGH                1
YTANPC                PJRCIN                1
BNSKRG                FWRJTA                1
GMKMFM                IWXGSV                1
uiop                  jlk                   1
FHKVQR                PFTGHO                1
MCKPFF                AWGWNT                1
YISEXD                GWIAGH                1
SVEOEA                UVYMJL                1
XOKIBU                RNTRWJ                1
AQGQHB                FQJUMG                1
AFYJXJ                UFXUIT                1
ASWOLI                HHWUTG                1
LOGPIW                GMASXE                1
CKXKNI                UWRHVT                1
XYRETY                PQUPBQ                1
JFBGKC                EMGONA                1
GUDCFA                BBRCJG                1
GUXVNE                LYFSWH                1
WXFCJL                BIKHYW                1
PFNVCX                UBRPXD                1

```

```

1 SELECT reader_name, reader_surname, count(*) AS ticket_count
2 FROM reader AS r INNER JOIN
3 (SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
4 AS rt ON r.reader_id = rt.reader_id
5 GROUP BY reader_name, reader_surname
6 ORDER BY ticket_count desc

```

Результат План выполн... Сообщения Notifications

| | reader_name name | reader_surname name | ticket_count bigint |
|----|---------------------|------------------------|------------------------|
| 1 | UHLPGV | LGWITA | 2 |
| 2 | HKDDPI | LEATXI | 2 |
| 3 | NCNYCB | SPONHJ | 1 |
| 4 | EBKDXV | HORMXV | 1 |
| 5 | UQPMMR | CBATGH | 1 |
| 6 | YTANPC | PJRCIN | 1 |
| 7 | BNSKRG | FWRJTA | 1 |
| 8 | GMKMFM | IWXGSV | 1 |
| 9 | uiop | jlk | 1 |
| 10 | FHKVQR | PFTGHO | 1 |
| 11 | MCKPFF | AWGWNT | 1 |
| 12 | YISEXD | GWIAGH | 1 |
| 13 | SVEOEA | UVYMJL | 1 |

```

1 EXPLAIN ANALYZE
2 SELECT reader_name, reader_surname, count(*) AS ticket_count
3 FROM reader AS r INNER JOIN
4 (SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
5 AS rt ON r.reader_id = rt.reader_id
6 GROUP BY reader_name, reader_surname
7 ORDER BY ticket_count desc

```

Результат План выполн... Сообщения Notifications

| | QUERY PLAN |
|----|--|
| 4 | HashAggregate (cost=61.91..67.36 rows=347 width=136) (actual time=0.217..0.222 rows=32 loops=1) |
| 5 | Group Key: r.reader_name, r.reader_surname |
| 6 | -> Hash Join (cost=33.31..70.45 rows=1528 width=128) (actual time=0.200..0.208 rows=34 loops=1) |
| 7 | Hash Cond: (reader_ticket.reader_id = r.reader_id) |
| 8 | -> Seq Scan on reader_ticket (cost=0.00..33.13 rows=1528 width=4) (actual time=0.009..0.012 rows=34 loops=1) |
| 9 | Filter: (date_issued > '2015-10-10'::date) |
| 10 | Rows Removed by Filter: 16 |
| 11 | -> Hash (cost=26.47..26.47 rows=547 width=132) (actual time=0.185..0.185 rows=547 loops=1) |
| 12 | Buckets: 1024 Batches: 1 Memory Usage: 96kB |
| 13 | -> Seq Scan on reader r (cost=0.00..26.47 rows=547 width=132) (actual time=0.006..0.090 rows=547 loops=1) |
| 14 | Planning Time: 0.215 ms |
| 15 | Execution Time: 0.271 ms |

Пошук найпопулярніших за певний період авторів

```
1. Add data
2. Delete data
3. Update data
4. Input random data
5. Search for books by author's name
6. Search for readers with the largest number of books in a given period
7. Search for the most popular authors for a certain period
8. View table
9. View table columns name
10. Exit
Enter the menu number: 7
Input date(YYYY-MM-DD): 2015-10-10
Крістофер                Паоліні                4
ВТНОЕМ                   DXCGCI                 3
АІННQY                   VULQFE                 3
Олександр                Панчін                 2
VFLWMJ                   FIIVNW                 2
KWEXCM                   RDQMEM                 2
Джоан                    Роулінг                2
NJLFJA                   LTWRKC                 1
USQHCG                   TFMNQT                 1
JOXPNB                   WVALMT                 1
Андре                    Хортон                 1
VJBWQD                   NOJNGR                 1
MKWVJW                   BDMYMF                 1
LJQTSC                   QTDEUC                 1
Карл                     Саган                  1
JBCBNA                   XXHHRJ                 1
XWDBFN                   LPUNLH                 1
GXDINH                   UMGCGY                 1
KORNXX                   YLIEKS                 1
Execution Time: 0.244 ms
```

Query Editor История запр...

```
1 SELECT author_name, author_surname, count(*) AS ticket_count
2 FROM author AS a INNER JOIN
3 (SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count
4 FROM book AS b INNER JOIN
5 (SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
6 AS rt ON b.book_id = rt.book_id
7 GROUP BY rt.book_id, b.book_name, author_id
8 ORDER BY ticket_count desc)
9 AS res ON res.author_id = a.author_id
10 GROUP BY author_name, author_surname
11 ORDER BY ticket_count desc
```

Результат План выполн... Сообщения Notifications

| | author_name name | author_surname name | ticket_count bigint | |
|---|---------------------|------------------------|------------------------|---|
| 1 | Крістофер | Паоліні | | 4 |
| 2 | ВТНОЕМ | DXCGCI | | 3 |
| 3 | АІННQY | VULQFE | | 3 |
| 4 | Олександр | Панчін | | 2 |
| 5 | VFLWMJ | FIIVNW | | 2 |
| 6 | KWEXCM | RDQMEM | | 2 |
| 7 | Джоан | Роулінг | | 2 |
| 8 | NJLFJA | LTWRKC | | 1 |

```

1  EXPLAIN ANALYZE
2  SELECT author_name, author_surname, count(*) AS ticket_count
3  FROM author AS a INNER JOIN
4  (SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count
5   FROM book AS b INNER JOIN
6   (SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
7   AS rt ON b.book_id = rt.book_id
8   GROUP BY rt.book_id, b.book_name, author_id
9   ORDER BY ticket_count desc)
10 AS res ON res.author_id = a.author_id
11 GROUP BY author_name, author_surname
12 ORDER BY ticket_count desc

```

Результат План выполн... Сообщения Notifications

| | QUERY PLAN | |
|----|---|--|
| | text | |
| 19 | Buckets: 1024 Batches: 1 Memory Usage: 16kB | |
| 20 | -> Seq Scan on book b (cost=0.00..3.56 rows=156 width=16) (actual time=0.007..0.020 rows=... | |
| 21 | -> Hash (cost=4.55..4.55 rows=55 width=132) (actual time=0.034..0.034 rows=55 loops=1) | |
| 22 | Buckets: 1024 Batches: 1 Memory Usage: 17kB | |
| 23 | -> Seq Scan on author a (cost=0.00..4.55 rows=55 width=132) (actual time=0.009..0.019 rows=55 loops=... | |
| 24 | Planning Time: 0.379 ms | |
| 25 | Execution Time: 0.211 ms | |

Завдання 4:

Controller.java:

```
1 package db.lab2.controller;
2
3 import db.lab2.model.Model;
4 import db.lab2.view.View;
5
6
7 public class Controller {
8     public static void main(String[] args) {
9         String db = "library";
10        String login = "postgres";
11        String password = "124/1/28";
12        Model.connectionInitialization(db, login, password);
13
14        menu();
15
16        Model.connectionClose();
17    }
18
19    public static void menu() {
20        int numberMenu = 0;
21        while(true) {
22            numberMenu = View.viewMenu();
23
24            switch(numberMenu) {
25                case 1: {
26                    String[] addRows = {"reader_id", "reader_name", "reader_surname", "reader_email"};
27                    String[] addData = View.inputAddData();
28                    String table = addData[addData.length - 1];
29                    View.performOperation(Model.addData(table, addRows, addData));
30                    break;
31                }
32                case 2:{
33                    String[] deleteData = View.inputDeleteData();
34                    View.performOperation(Model.deleteData(deleteData[0], deleteData[1], deleteData[2]));
35                    break;
36                }
37                case 3:{
38                    String[] updateRows = {"reader_id", "reader_name", "reader_surname", "reader_email"};
39                    String[] updateData = View.inputUpdateData();
40                    int length = updateData.length;
41                    View.performOperation(Model.updateData(updateData[length - 3], updateRows, updateData, updateData[length - 2], updateData[length - 1]));
42                    break;
43                }
44                case 4:{
45                    String[] randomData = View.inputRandomData();
46                    View.performOperation(Model.randomData(randomData[0], Integer.parseInt(randomData[1])));
47                    break;
48                }
49                case 5:{
50                    String searchName = View.inputSearchByName();
51                    Model.searchByName(searchName);
52                    break;
53                }
54                case 6:{
55                    String date = View.inputCountBookInReader();
56                    Model.countBookInReader(date);
57                    break;
58                }
59                case 7:{
60                    String date = View.inputCountPopularAuthor();
61                    Model.countPopularAuthor(date);
62                    break;
63                }
64                case 8:{
65                    String table = View.inputTable();
66                    Model.printTable(table);
67                    break;
68                }
69                case 9:{
70                    String table = View.inputTable();
71                    Model.printColumnsNameType(table);
72                    break;
73                }
74                case 10: {
75                    return;
76                }
77            }
78        }
79    }
80 }
81 }
```

Model.java:

```
1  package db.lab2.model;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8  import db.lab2.view.View;
9
10
11  public class Model {
12      private static Connection connection = null;
13      private static String database = null;
14
15      public static void connectionInitialization(String db, String login, String password) {
16          try {
17              database = db;
18              Class.forName("org.postgresql.Driver");
19              String url = "jdbc:postgresql://localhost:5432/" + database;
20              connection = DriverManager.getConnection(url, login, password);
21
22              if(connection != null) {
23                  System.out.println("The connection to the database is established\n");
24              } else {
25                  System.out.println("No connection to database\n");
26                  return;
27              }
28          } catch (Exception e) {
29              System.out.println(e);
30          }
31      }
32
33
34      public static void connectionClose() {
35          try {
36              connection.close();
37          } catch (SQLException e) {
38              postgresError(e);
39          }
40      }
41
42
43      public static boolean addData(String table, String[] columns, String[] data) {
44          try {
45              int numberColumn = countColumns(table);
46              String insertData = "INSERT INTO " + table + " (";
47              for(int i = 0; i < numberColumn; i++) {
48                  insertData += columns[i];
49                  if(i != numberColumn - 1) {
50                      insertData += ",";
51                  }
52              }
53              insertData += ") VALUES (";
54              for(int i = 0; i < numberColumn; i++) {
55                  insertData += "'" + data[i] + "'";
56                  if(i != numberColumn - 1) {
57                      insertData += ",";
58                  }
59              }
60              insertData += ")";
61
62              Statement statement = connection.createStatement();
63              statement.executeUpdate(insertData);
64
65              statement.close();
66          } catch (SQLException e) {
67              postgresError(e);
68              return false;
69          }
70          return true;
71      }
72
73
74      public static boolean deleteData(String table, String row, String value) {
75          try {
76              String deleteData;
77              if(row == null) deleteData = "DELETE FROM " + table;
78              else deleteData = "DELETE FROM " + table + " WHERE " + row + " = " + "'" + value + "'";
79
80              Statement statement = connection.createStatement();
81              statement.executeUpdate(deleteData);
82
83              statement.close();
84          } catch (SQLException e) {
85              postgresError(e);
86              return false;
87          }
88          return true;
89      }
90
91  }
```



```

91     public static boolean updateData(String table, String[] newRow, String[] newData, String oldRow, String oldData) {
92         try {
93             int numberColumns = countColumns(table);
94             String updateData = "UPDATE " + table + " SET ";
95             for(int i = 0; i < numberColumns; i++) {
96                 updateData += newRow[i] + " = " + "" + newData[i] + """;
97                 if(i != numberColumns - 1) {
98                     updateData += ",";
99                 }
100             }
101
102             updateData += " WHERE " + oldRow + " = " + "" + oldData + """;
103             Statement statement = connection.createStatement();
104             statement.executeUpdate(updateData);
105
106             statement.close();
107         }catch (SQLException e) {
108             postgresError(e);
109             return false;
110         }
111         return true;
112     }
113
114     public static boolean randomData(String table, int numberColumns) {
115         switch(table) {
116             case "reader":
117                 return randomReader(numberColumns);
118             case "author":
119                 return randomAuthor(numberColumns);
120             case "book":
121                 return randomBook(numberColumns);
122             case "reader_ticket":
123                 return randomReaderTicket(numberColumns);
124             case "all":
125                 randomReader(numberColumns);
126                 randomAuthor(numberColumns);
127                 randomBook(numberColumns);
128                 return randomReaderTicket(numberColumns);
129             default:
130                 return false;
131         }
132     }
133
134     private static boolean randomReader(int numberColumns) {
135         String randomReader = "INSERT INTO reader (reader_id, reader_name, reader_surname, reader_email)"
136             + "SELECT trunc(10 + random()*1000000)::int, "
137             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
138             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
139             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
140             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
141             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
142             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
143             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
144             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
145             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
146             + "FROM generate_series(1, " + numberColumns + ")";
147
148         try {
149             Statement statement = connection.createStatement();
150             statement.executeUpdate(randomReader);
151             statement.close();
152         }catch (SQLException e) {
153             postgresError(e);
154             return false;
155         }
156         return true;
157     }
158
159     private static boolean randomAuthor(int numberColumns) {
160         String randomAuthor = "INSERT INTO author (author_id, author_name, author_surname, author_email)"
161             + "SELECT trunc(10 + random()*1000000)::int, "
162             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
163             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
164             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
165             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
166             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
167             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),"
168             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||"
169             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
170             + "FROM generate_series(1, " + numberColumns + ")";
171
172         try {
173             Statement statement = connection.createStatement();
174             statement.executeUpdate(randomAuthor);
175             statement.close();
176         }catch (SQLException e) {
177             postgresError(e);
178             return false;
179         }
180         return true;
181     }
182 }

```

```

182     private static boolean randomBook(int numberColumns) {
183         String randomBook = "INSERT INTO book (book_id, book_name, author_id, year_publication, count_books)\r\n"
184             + "(SELECT trunc(random()*10000)::int,\r\n"
185             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||\r\n"
186             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||\r\n"
187             + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),\r\n"
188             + "author_id,\r\n"
189             + "trunc(1960 + random()*150)::int,\r\n"
190             + "trunc(random()*10000)::int\r\n"
191             + "FROM author tablesample BERNOULLI(100)\r\n"
192             + "ORDER BY random()) LIMIT " + numberColumns;
193
194         try {
195             Statement statement = connection.createStatement();
196             statement.executeUpdate(randomBook);
197             statement.close();
198         } catch (SQLException e) {
199             postgresError(e);
200             return false;
201         }
202         return true;
203     }
204
205     private static boolean randomReaderTicket(int numberColumns) {
206         String randomReaderTicket = "INSERT INTO reader_ticket (reader_id, book_id, date_issued, validity)\r\n"
207             + "(SELECT reader_id, book_id,\r\n"
208             + "date '2012-01-01' + trunc(random() * 100 * (date '2020-05-20' - date '2012-01-01'))::int,\r\n"
209             + "trunc(15 + random()*10000)::int\r\n"
210             + "FROM reader, book tablesample BERNOULLI(100)\r\n"
211             + "ORDER BY random()) LIMIT " + numberColumns;
212
213         try {
214             Statement statement = connection.createStatement();
215             statement.executeUpdate(randomReaderTicket);
216             statement.close();
217         } catch (SQLException e) {
218             postgresError(e);
219             return false;
220         }
221         return true;
222     }
223
224     // return books by author name
225     public static void searchByName(String name) {
226         String joinText = "SELECT a.author_name, a.author_surname, book_name\r\n"
227             + "FROM book AS b INNER JOIN\r\n"
228             + "(SELECT author_id, author_name, author_surname \r\n"
229             + "FROM author WHERE author_name LIKE '" + name + "') \r\n"
230             + "AS a ON b.author_id = a.author_id";
231
232         String explainJoinText = "EXPLAIN ANALYZE (" + joinText + ")";
233
234         try {
235             Statement statement = connection.createStatement();
236             ResultSet rs = statement.executeQuery(joinText);
237             View.printTable(rs, 3);
238             rs = statement.executeQuery(explainJoinText);
239             View.printExplain(rs);
240             statement.close();
241             rs.close();
242         } catch (SQLException e) {
243             postgresError(e);
244         }
245     }
246
247     // return number book, which reader take for a certain period
248     public static void countBookInReader(String date) {
249         String countBookInReader = "SELECT reader_name, reader_surname, count(*) AS ticket_count \r\n"
250             + "FROM reader AS r INNER JOIN\r\n"
251             + "(SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '" + date + "')\r\n"
252             + "AS rt ON r.reader_id = rt.reader_id\r\n"
253             + "GROUP BY reader_name, reader_surname\r\n"
254             + "ORDER BY ticket_count desc";
255
256         String explainCountBookInReader = "EXPLAIN ANALYZE (" + countBookInReader + ")";
257
258         try {
259             Statement statement = connection.createStatement();
260             ResultSet rs = statement.executeQuery(countBookInReader);
261             View.printTable(rs, 3);
262             rs = statement.executeQuery(explainCountBookInReader);
263             View.printExplain(rs);
264             statement.close();
265             rs.close();
266         } catch (SQLException e) {
267             postgresError(e);
268         }
269     }
270 }

```

```

265 // return the most popular authors and their books
266 public static void countPopularAuthor(String date) {
267     String countPopularAuthor = "SELECT author_name, author_surname, count(*) AS ticket_count\r\n"
268         + "FROM author AS a INNER JOIN\r\n"
269         + "(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count\r\n"
270         + "FROM book AS b INNER JOIN\r\n"
271         + "(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')\r\n"
272         + "AS rt ON b.book_id = rt.book_id\r\n"
273         + "GROUP BY rt.book_id, b.book_name, author_id\r\n"
274         + "ORDER BY ticket_count desc)\r\n"
275         + "AS res ON res.author_id = a.author_id\r\n"
276         + "GROUP BY author_name, author_surname\r\n"
277         + "ORDER BY ticket_count desc";
278     String explainCountPopularAuthor = "EXPLAIN ANALYZE (" + countPopularAuthor + ")";
279     try {
280         Statement statement = connection.createStatement();
281         ResultSet rs = statement.executeQuery(countPopularAuthor);
282         View.printTable(rs, 3);
283         rs = statement.executeQuery(explainCountPopularAuthor);
284         View.printExplain(rs);
285         statement.close();
286         rs.close();
287     }catch (SQLException e) {
288         postgresError(e);
289     }
290 }
291
292 // return the most popular authors and their books
293 public static void countPopularAuthor(String date) {
294     String countPopularAuthor = "SELECT author_name, author_surname, count(*) AS ticket_count\r\n"
295         + "FROM author AS a INNER JOIN\r\n"
296         + "(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count\r\n"
297         + "FROM book AS b INNER JOIN\r\n"
298         + "(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')\r\n"
299         + "AS rt ON b.book_id = rt.book_id\r\n"
300         + "GROUP BY rt.book_id, b.book_name, author_id\r\n"
301         + "ORDER BY ticket_count desc)\r\n"
302         + "AS res ON res.author_id = a.author_id\r\n"
303         + "GROUP BY author_name, author_surname\r\n"
304         + "ORDER BY ticket_count desc";
305     String explainCountPopularAuthor = "EXPLAIN ANALYZE (" + countPopularAuthor + ")";
306     try {
307         Statement statement = connection.createStatement();
308         ResultSet rs = statement.executeQuery(countPopularAuthor);
309         View.printTable(rs, 3);
310         rs = statement.executeQuery(explainCountPopularAuthor);
311         View.printExplain(rs);
312         statement.close();
313         rs.close();
314     }catch (SQLException e) {
315         postgresError(e);
316     }
317 }
318
319 public static String[] getColumnsName(String table) {
320     String[] result = null;
321     try {
322         String getColumns = "SELECT column_name FROM information_schema.columns WHERE table_name = '" + table + "'";
323
324         Statement statement = connection.createStatement();
325         ResultSet rs = statement.executeQuery(getColumns);
326         int size = countColumns(table);
327         result = new String[size];
328
329         for(int i = 0; rs.next(); i++) {
330             result[i] = rs.getString(1);
331         }
332         statement.close();
333         rs.close();
334     }catch (SQLException e) {
335         postgresError(e);
336         return null;
337     }
338
339     return result;
340 }

```

```

316     public static String[] getColumnsType(String table) {
317         String[] result = null;
318         try {
319             String getType = "SELECT data_type FROM information_schema.columns WHERE table_name = '" + table + "'";
320
321             Statement statement = connection.createStatement();
322             ResultSet rs = statement.executeQuery(getType);
323
324             result = new String[countColumns(table)];
325
326             for(int i = 0; rs.next(); i++) {
327                 result[i] = rs.getString(1);
328             }
329
330             statement.close();
331             rs.close();
332         }catch (SQLException e) {
333             postgresError(e);
334             return null;
335         }
336
337         return result;
338     }
339
340     public static int countColumns(String table) {
341         int result = 0;
342
343         String countColumns = "SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_catalog = '" +
344             database + "' AND table_name = '" + table + "'";
345
346         try {
347             Statement statement = connection.createStatement();
348             ResultSet rs = statement.executeQuery(countColumns);
349             if(rs.next()) result = rs.getInt(1);
350             statement.close();
351             rs.close();
352         }catch (SQLException e) {
353             postgresError(e);
354             return 0;
355         }
356         return result;
357     }
358
359     public static void printTable(String table) {
360         try {
361             String printTable = "SELECT * FROM " + table;
362             Statement statement = connection.createStatement();
363             ResultSet rs = statement.executeQuery(printTable);
364             int size = countColumns(table);
365             View.printTable(rs, size);
366             statement.close();
367             rs.close();
368         }catch (SQLException e) {
369             postgresError(e);
370         }
371     }
372
373     private static void postgresError(SQLException e) {
374         System.out.println("\nError: Incorrect data entered");
375     }
376
377     public static void printColumnsNameType(String table) {
378         String[] columnsName = getColumnsName(table);
379         String[] columnsType = getColumnsType(table);
380
381         int size = columnsName.length;
382
383         String[][] columnsNameType = new String[size][2];
384
385         for(int i = 0; i < size; i++) {
386             columnsNameType[i][0] = columnsName[i];
387             columnsNameType[i][1] = columnsType[i];
388         }
389
390         View.printMatrix(columnsNameType);
391     }
392 }

```

View.java

```
1 package db.lab2.view;
2
3 import db.lab2.model.Model;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.Scanner;
7
8
9
10 public class View {
11     public static void printTable(ResultSet rs, int size) {
12         try {
13             while (rs.next()) {
14                 for(int i = 1; i <= size; i++) {
15                     System.out.printf("%-50s", rs.getString(i));
16                 }
17                 System.out.printf("\n");
18             }
19         } catch (Exception e) {
20             System.out.println(e);
21         }
22     }
23
24     public static void printExplain(ResultSet rs) {
25         try {
26             String time;
27             while(rs.next()) {
28                 if((time = rs.getString(1)).startsWith("Execution Time:")) {
29                     System.out.println(time);
30                 }
31             }
32         } catch (SQLException e) {
33             System.out.println(e);
34         }
35     }
36
37     public static void performOperation(boolean bool) {
38         if(bool) {
39             System.out.println("\nOperation is performed");
40         } else {
41             System.out.println("\nOperation is not performed");
42         }
43     }
44
45     public static int viewMenu() {
46         String menu = "1. Add data\n"
47             + "2. Delete data\n"
48             + "3. Update data\n"
49             + "4. Input random data\n"
50             + "5. Search for books by author's name\n"
51             + "6. Search for readers with the largest number of books in a given period\n"
52             + "7. Search for the most popular authors for a certain period\n"
53             + "8. View table\n"
54             + "9. View table columns name\n"
55             + "10. Exit";
56
57         Scanner in = new Scanner(System.in);
58         System.out.println("\n" + menu);
59         System.out.print("Enter the menu number: ");
60         int numberMenu = in.nextInt();
61
62         return numberMenu;
63     }
64
65     public static String[] inputAddData() {
66         Scanner in = new Scanner(System.in);
67
68         System.out.print("Input table: ");
69         String table = in.nextLine();
70
71         int numberColumns = Model.countColumns(table);
72         String[] result = new String[numberColumns + 1];
73
74         String[] columnsName = Model.getColumnsName(table);
75
76         for(int i = 0; i < numberColumns; i++) {
77             System.out.print("Input " + columnsName[i] + ": ");
78             result[i] = in.nextLine();
79         }
80
81         result[numberColumns] = table;
82
83         return result;
84     }
85 }
```

```

86     public static String[] inputDeleteData() {
87         Scanner in = new Scanner(System.in);
88
89         String[] result = new String[3];
90
91         System.out.print("Input table: ");
92         result[0] = in.nextLine();
93
94         System.out.print("Input the name of the column by which the data will be displayed(if delete all data in table input 'null'): ");
95         result[1] = in.nextLine();
96
97         System.out.print("Input value column: ");
98         result[2] = in.nextLine();
99
100        return result;
101    }
102
103    public static String[] inputUpdateData() {
104        Scanner in = new Scanner(System.in);
105
106        System.out.print("Input table: ");
107        String table = in.nextLine();
108
109        int numberColumns = Model.countColumns(table);
110        String[] result = new String[numberColumns + 3];
111
112        String[] columnsName = Model.getColumnsName(table);
113
114        result[numberColumns] = table;
115
116        System.out.print("Input the name of the column in which the data changes: ");
117        result[numberColumns + 1] = in.nextLine();
118
119        System.out.print("Input value column: ");
120        result[numberColumns + 2] = in.nextLine();
121
122        for(int i = 0; i < numberColumns; i++) {
123            System.out.print("Input " + columnsName[i] + ": ");
124            result[i] = in.nextLine();
125        }
126
127        return result;
128    }
129
130    public static String[] inputRandomData() {
131        Scanner in = new Scanner(System.in);
132
133        String[] result = new String[2];
134
135        System.out.print("Input table( all tables, input 'all'): ");
136        result[0] = in.nextLine();
137
138        System.out.print("Input the amount of random data: ");
139        result[1] = in.nextLine();
140
141        return result;
142    }
143
144    public static String inputSearchByName() {
145        Scanner in = new Scanner(System.in);
146
147        System.out.print("Input name: ");
148        String name = in.nextLine();
149
150        return name;
151    }
152
153    public static String inputCountBookInReader() {
154        Scanner in = new Scanner(System.in);
155
156        System.out.print("Input date(YYYY-MM-DD): ");
157        String date = in.nextLine();
158
159        return date;
160    }
161

```

```
162     public static String inputCountPopularAuthor() {
163         Scanner in = new Scanner(System.in);
164
165         System.out.print("Input date(YYYY-MM-DD): ");
166         String date = in.nextLine();
167
168         return date;
169     }
170
171     public static String inputTable() {
172         Scanner in = new Scanner(System.in);
173
174         System.out.print("Input table: ");
175         String table = in.nextLine();
176
177         return table;
178     }
179
180     public static void printMatrix(String[][] matrix) {
181         int numberOfRows = matrix.length;
182         int numberOfColumns = matrix[0].length;
183
184         for(int i = 0; i < numberOfRows; i++) {
185             for(int j = 0; j < numberOfColumns; j++) {
186                 System.out.printf("%-20s", matrix[i][j]);
187             }
188             System.out.println();
189         }
190     }
191 }
```