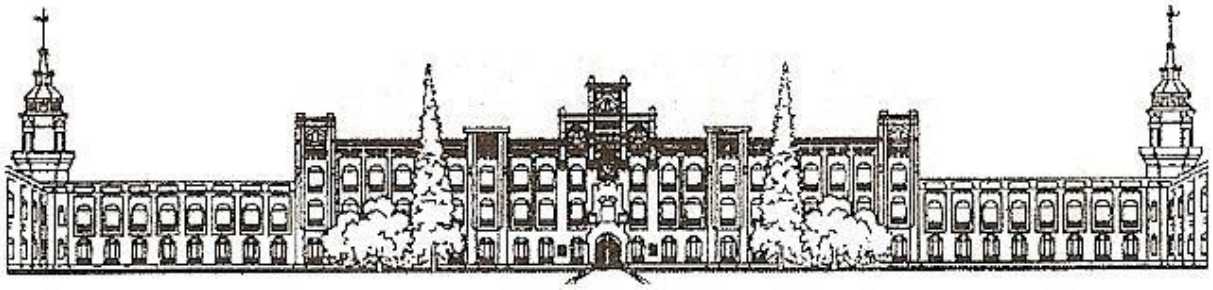


Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікрського»

Лабораторна робота № 2

З дисципліни «Бази даних і засоби управління»
Назва: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав студент групи: КВ-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                                   Деп                                     jd@jd.com
6                                       Кіану                                   Рівз                                    kr@kr.com
9                                       qwertty                                 asdfggh                                 zxcvbbn
11                                      qwerty                                  asdfgh                                  zxcvbn
474                                     NCCWLN                                  UKBRYA                                  OMYSAX
937                                     PFPATA                                  HEQSWR                                  ODOORA
482                                     CMLEJK                                  KIBOGH                                  KJJUTX
142                                     EAQVHB                                  BYLPSL                                  QQNUJL
756                                     OQPGHI                                  NKWFAB                                  NILHSX
889                                     ASCQSI                                  TWUJXB                                  FRBSCN
952                                     VBXTWO                                  RGYJIO                                  WRFNPQ
111                                     WSOEAI                                  WFPAKJ                                  FEHRIO
786                                     EBKDXV                                  HORMXV                                  DTTNWB
698                                     XNWWOO                                  PJTQAL                                  QFGUHI
1007                                    BHYKOU                                  UEKKVL                                  KTSORP
799                                     YDPYCL                                  LJOESG                                  KMKFRC
908                                     FALFHU                                  KADYPJ                                  CLVWHN
68                                      ILTJGW                                  LNNROS                                  MSNWXW
210                                     EHYOXF                                  THYSKA                                  MUQKBH
136                                     FLGNKI                                  WUFJQV                                  NODYBR
399                                     GUDCFA                                  BBRCJG                                  BFLJQX
130                                     FRAIQQ                                  NCRMEQ                                  AIDHWK
567                                     NYXERF                                  CMJUUJ                                  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                 Йоханссон               sj@sg.com
5                                   Джоні                   Деп                     jd@jd.com
6                                   Кіану                   Рівз                    kr@kr.com
9                                   qwertty                 asdfggh                 zxcvbbn
10                                  qazwsx                  edcrfv                  tgbyhn
11                                  qwerty                  asdfgh                  zxcvbn
474                                 NCCWLN                  UKBRYA                  OMYSAX
937                                 PFPATA                  HEQSWR                  ODOORA
482                                 CMLEJK                  KIBOGH                  KJJUTX
142                                 EAQVHB                  BYLPSL                  QQNUJL
756                                 OQPGHI                  NKWFAB                  NILHSX
889                                 ASCQSI                  TWUJXB                  FRBSCN
952                                 VBXTWO                  RGYJIO                  WRFNPQ
111                                 WSOEAI                  WFPAKJ                  FEHRIO
786                                 EBKDXV                  HORMXV                  DTTNWB
698                                 XNWWOO                  PJTQAL                  QFGUHI
1007                                BHYKOU                  UEKKVL                  KTSORP
799                                 YDPYCL                  LJOESG                  KMKFRC
908                                 FALFHU                  KADYPJ                  CLVWHN
68                                  ILTJGW                  LNNROS                  MSNWXW
210                                 EHYOXF                  THYSKA                  MUQKBH
136                                 FLGNKI                  WUFJQV                  NODYBR
399                                 GUDCFA                  BBRCJG                  BFLJQX
130                                 FRAIQQ                  NCRMEQ                  AIDHWK
567                                 NYXERF                  CMJUUJ                  LJUWAL
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                                   Рівз                                    kr@kr.com
9                                       qwertty                                 asdfggh                                 zxcvbbn
11                                      qwerty                                  asdfgh                                  zxcvbn
474                                     NCCWLN                                  UKBRYA                                  OMYSAX
937                                     PFPATA                                  HEQSWR                                  ODOORA
482                                     CMLEJK                                  KIBOGH                                  KJJUTX
142                                     EAQVHB                                  BYLPSL                                  QQNUJL
756                                     OQPGHI                                  NKWFAB                                  NILHSX
889                                     ASCQSI                                  TWUJXB                                  FRBSCN
952                                     VBXTWO                                  RGYJIO                                  WRFNPQ
111                                     WSOEAI                                  WFPAKJ                                  FEHRIO
786                                     EBKDXV                                  HORMXV                                  DTTNWB
698                                     XNWWOO                                  PJTQAL                                  QFGUHI
1007                                    BHYKOU                                  UEKKVL                                  KTSORP
799                                     YDPYCL                                  LJOESG                                  KMKFRC
908                                     FALFHU                                  KADYPJ                                  CLVWHN
68                                      ILTJGW                                  LNNROS                                  MSNWXW
210                                     EHYOXF                                  THYSKA                                  MUQKBH
136                                     FLGNKI                                  WUFJQV                                  NODYBR
399                                     GUDCFA                                  BBRCJG                                  BFLJQX
130                                     FRAIQQ                                  NCRMEQ                                  AIDHWK
567                                     NYXERF                                  CMJUUJ                                  LJUWAL
602                                     GAMXSJ                                  MBTQYA                                  EYVKCN
```

Обробка помилок:

Видалення даних з таблиці reader, при цьому у підлеглій 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                             Йоханссон                           sj@sg.com
5                                   Джоні                               Деп                                 jd@jd.com
6                                   Кіану                               Рівз                                kr@kr.com
9                                   qwertty                             asdfggh                             zxcvbbn
474                                 NCCWLN                              UKBRYA                              OMYSAX
937                                 PFPATA                              HEQSWR                              ODOORA
482                                 CMLEJK                              KIBOGH                              KJJUTX
142                                 EAQVHB                              BYLPSL                              QQNUJL
756                                 OQPGHI                              NKWFAB                              NILHSX
889                                 ASCQSI                              TWUJXB                              FRBSCN
952                                 VBXTWO                              RGYJIO                              WRFNPQ
111                                 WSOEAI                              WFPAKJ                              FEHRIO
786                                 EBKDXV                              HORMXV                              DTTNWB
698                                 XNWWOO                              PJTQAL                              QFGUHI
1007                                BHYKOU                              UEKKVL                              KTSORP
799                                 YDPYCL                              LJOESG                              KMKFRC
908                                 FALFHU                              KADYPJ                              CLVWHN
68                                  ILTJGW                              LNNROS                              MSNWXW
210                                 EHYOXF                              THYSKA                              MUQKBH
136                                 FLGNKI                              WUFJQV                              NODYBR
399                                 GUDCFA                              BBRCJG                              BFLJQX
130                                 FRAIQQ                              NCRMEQ                              AIDHWK
567                                 NYXERF                              CMJUUJ                              LJUWAL
602                                 GAMXSJ                              MBTQYA                              EYVKCN
10                                  uiop                                jlk                                 m
```

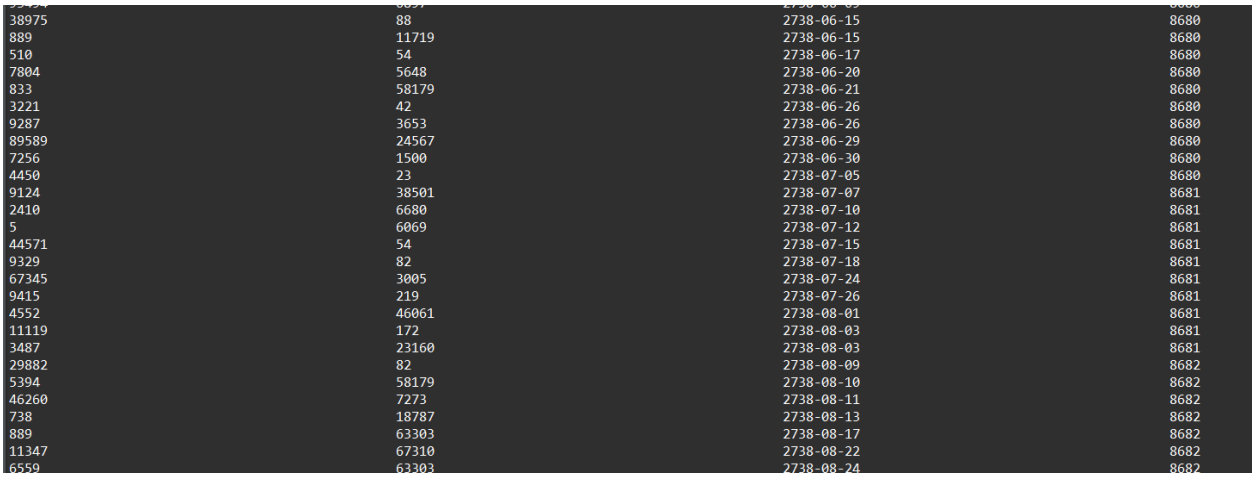
Обробка помилок:

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	2738-06-15	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

	count bigint
1	100000

```sql
INSERT INTO book (book_id, book_name, author_id, year_publication, count_books)
(SELECT trunc(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),
author_id,
trunc(1960 + random()*150)::int,
trunc(random()*10000)::int
FROM author tablesample BERNOULLI(100)
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

```
ramdomReader = "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) ||"
 + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
 + "FROM generate_series(1, " + numberColumns + ")";
```

```
ramdomAuthor = "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) ||"
 + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
 + "FROM generate_series(1, " + numberColumns + ")";
```

```
ramdomBook = "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;
```

```
ramdomReaderTicket = "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 філосовський камінь
Джоан                                   Роулінг                                 CCCCCC
Джоан                                   Роулінг                                 CCCCCC
Джоан                                   Роулінг                                 BBBBBB
Джоан                                   Роулінг                                 EEEEEE
Джоан                                   Роулінг                                 BBBBBB
Джоан                                   Роулінг                                 TTTTTT
Джоан                                   Роулінг                                 RRRRRR
Execution Time: 0.083 ms
```

```sql
SELECT a.author_name, a.author_surname, book_name
FROM book  AS b INNER JOIN
(SELECT author_id, author_name, author_surname
FROM author WHERE author_name LIKE 'Джоан')
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
```

```sql
SELECT reader_name, reader_surname, count(*) AS ticket_count
FROM reader AS r INNER JOIN
(SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
AS rt ON r.reader_id = rt.reader_id
GROUP BY reader_name, reader_surname
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

```sql
EXPLAIN ANALYZE
SELECT reader_name, reader_surname, count(*) AS ticket_count
FROM reader AS r INNER JOIN
(SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
AS rt ON r.reader_id = rt.reader_id
GROUP BY reader_name, reader_surname
ORDER BY ticket_count desc
```

Результат | План выполн... | Сообщения | Notifications

	QUERY PLAN text
4	-> HashAggregate (cost=81.91..87.38 rows=547 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...
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 loop...
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
BTHOEM                                   DXCGCI                                   3
AINHQY                                   VULQFE                                   3
Олександр                                Панчін                                   2
VFLWMJ                                   FIIVNW                                   2
KWEXCM                                   RDQMEM                                   2
Джоан                                    Роулінг                                  2
NJLFJA                                   LTWRKC                                   1
USQHCG                                   TFMNQT                                   1
JOXPNB                                   WVALMT                                   1
Андре                                    Нортон                                   1
VJBWQD                                   NOJNGR                                   1
MKWVJW                                   BDMYMF                                   1
LJQTSC                                   QTDEUC                                   1
Карл                                     Саган                                    1
JBCBNA                                   XXHHRJ                                   1
XWDBFN                                   LPUNLH                                   1
GXDINH                                   UMCGCY                                   1
KORNXX                                   YLIEKS                                   1
Execution Time: 0.244 ms
```

Query Editor История запр...

```sql
SELECT author_name, author_surname, count(*) AS ticket_count
FROM author AS a INNER JOIN
(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count
FROM book AS b INNER JOIN
(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
AS rt ON b.book_id = rt.book_id
GROUP BY rt.book_id, b.book_name, author_id
ORDER BY ticket_count desc)
AS res ON res.author_id = a.author_id
GROUP BY author_name, author_surname
ORDER BY ticket_count desc
```

Результат План выполн... Сообщения Notifications

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

```sql
EXPLAIN ANALYZE
SELECT author_name, author_surname, count(*) AS ticket_count
FROM author AS a INNER JOIN
(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count
FROM book AS b INNER JOIN
(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')
AS rt ON b.book_id = rt.book_id
GROUP BY rt.book_id, b.book_name, author_id
ORDER BY ticket_count desc)
AS res ON res.author_id = a.author_id
GROUP BY author_name, author_surname
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:

```java
package db.lab2.controller;

import db.lab2.model.Model;
import db.lab2.view.View;


public class Controller {
        public static void main(String[] args) {
                String db = "library";
                String login = "postgres";
                String password = "124/1/28";
                Model.connectionInitialization(db, login, password);

                menu();

                Model.connectionClose();
        }


          public static void menu() {
                  int numberMenu = 0;
                  while(true) {
                          numberMenu = View.viewMenu();

                          switch(numberMenu) {
                          case 1: {
                                  String[] addRows = {"reader_id", "reader_name", "reader_surname", "reader_email"};
                                  String[] addData = View.inputAddData();
                                  String table = addData[addData.length - 1];
                                  View.performOperation(Model.addData(table, addRows, addData));
                                  break;
                          }
                          case 2:{
                                  String[] deleteData = View.inputDeleteData();
                                  View.performOperation(Model.deleteData(deleteData[0], deleteData[1], deleteData[2]));
                                  break;
                          }
                          case 3:{
                                  String[] updateRows = {"reader_id", "reader_name", "reader_surname", "reader_email"};
                                  String[] updateData = View.inputUpdateData();
                                  int length = updateData.length;
                                  View.performOperation(Model.updateData(updateData[length - 3], updateRows, updateData, updateData[length - 2], updateData[length - 1]));
                                  break;
                          }
                          case 4:{
                                  String[] randomData = View.inputRandomData();
                                  View.performOperation(Model.randomData(randomData[0], Integer.parseInt(randomData[1])));
                                  break;
                          }
                          case 5:{
                                  String searchName = View.inputSearchByName();
                                  Model.searchByName(searchName);
                                  break;
                          }
                          case 6:{
                                  String date = View.inputCountBookInReader();
                                  Model.countBookInReader(date);
                                  break;
                          }
                          case 7:{
                                  String date = View.inputCountPopularAuthor();
                                  Model.countPopularAuthor(date);
                                  break;
                          }
                          case 8:{
                                  String table = View.inputTable();
                                  Model.printTable(table);
                                  break;
                          }
                          case 9:{
                                  String table = View.inputTable();
                                  Model.printColumnsNameType(table);
                                  break;
                          }
                          case 10: {
                                  return;
                          }

                          }
                  }
          }
}
```

Model.java:

```java
package db.lab2.model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import db.lab2.view.View;


public class Model {
	private static Connection connection = null;
	private static String database = null;

	public static void connectionInitialization(String db, String login, String password) {
		try {
			database = db;
			Class.forName("org.postgresql.Driver");
			String url = "jdbc:postgresql://localhost:5432/" + database;
			connection = DriverManager.getConnection(url, login, password);

			if(connection != null) {
				System.out.println("The connection to the database is established\n");
			} else {
				System.out.println("No connection to database\n");
				return;
			}
		}catch (Exception e) {
				System.out.println(e);
		}
	}


	public static void connectionClose() {
		try {
			connection.close();
		}catch (SQLException e) {
			postgresError(e);
		}
	}

	public static boolean addData(String table, String[] columns, String[] data) {
		try {
			int numberColumn = countColumns(table);
			String insertData = "INSERT INTO " + table + " (";
			for(int i = 0; i < numberColumn; i++) {
				insertData += columns[i];
				if(i != numberColumn - 1) {
					insertData += ",";
				}
			}
			insertData += ") VALUES (";
			for(int i = 0; i < numberColumn; i++) {
				insertData += "'" + data[i] + "'";
				if(i != numberColumn - 1) {
					insertData += ",";
				}
			}
			insertData += ")";

			Statement statement = connection.createStatement();
			statement.executeUpdate(insertData);

			statement.close();
		}catch (SQLException e) {
			postgresError(e);
			return false;
		}
		return true;
	}

	public static boolean deleteData(String table, String row, String value) {
		try {
			String deleteData;
			if(row == null) deleteData = "DELETE FROM " + table;
			else deleteData = "DELETE FROM " + table + " WHERE " + row + " = " + "'" + value + "'";

			Statement statement = connection.createStatement();
			statement.executeUpdate(deleteData);

			statement.close();
		}catch (SQLException e) {
			postgresError(e);
			return false;
		}

		return true;
	}

```

```java
    public static boolean updateData(String table, String[] newRow, String[] newData, String oldRow, String oldData) {
            try {
                    int numberCoulumns = countColumns(table);
                    String updateData = "UPDATE " + table + " SET ";
                    for(int i = 0; i < numberCoulumns; i++) {
                            updateData += newRow[i] + " = " + "'" + newData[i] + "'";
                            if(i != numberCoulumns - 1) {
                                    updateData += ",";
                            }
                    }

            updateData += " WHERE " + oldRow + " = " + "'" + oldData + "'";
            Statement statement = connection.createStatement();
            statement.executeUpdate(updateData);

            statement.close();
            }catch (SQLException e) {
                    postgresError(e);
                    return false;
            }
            return true;
    }
    public static boolean randomData(String table, int numberColumns) {
            switch(table) {
            case "reader":
                    return ramdomReader(numberColumns);
            case "author":
                    return ramdomAuthor(numberColumns);
            case "book":
                    return ramdomBook(numberColumns);
            case "reader_ticket":
                    return ramdomReaderTicket(numberColumns);
            case "all":
                    ramdomReader(numberColumns);
                    ramdomAuthor(numberColumns);
                    ramdomBook(numberColumns);
                    return ramdomReaderTicket(numberColumns);
            default:
                    return false;
            }
    }
    private static boolean ramdomReader(int numberColumns) {
            String ramdomReader = "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) ||"
                            + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
                            + "FROM generate_series(1, " + numberColumns + ")";
            try {
                    Statement statement = connection.createStatement();
                    statement.executeUpdate(ramdomReader);
                    statement.close();
            }catch (SQLException e) {
                    postgresError(e);
                    return false;
            }
            return true;
    }
    private static boolean ramdomAuthor(int numberColumns) {
            String ramdomAuthor = "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) ||"
                            + "chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)"
                            + "FROM generate_series(1, " + numberColumns + ")";
            try {
                    Statement statement = connection.createStatement();
                    statement.executeUpdate(ramdomAuthor);
                    statement.close();
            }catch (SQLException e) {
                    postgresError(e);
                    return false;
            }
            return true;
    }
```

```java
    private static boolean ramdomBook(int numberColumns) {
        String ramdomBook = "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;
        try {
            Statement statement = connection.createStatement();
            statement.executeUpdate(ramdomBook);
            statement.close();
        }catch (SQLException e) {
            postgresError(e);
            return false;
        }
        return true;
    }

    private static boolean ramdomReaderTicket(int numberColumns) {
        String ramdomReaderTicket = "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;
        try {
            Statement statement = connection.createStatement();
            statement.executeUpdate(ramdomReaderTicket);
            statement.close();
        }catch (SQLException e) {
            postgresError(e);
            return false;
        }
        return true;
    }

    // return books by author name
    public static void searchByName(String name) {
        String joinText = "SELECT a.author_name, a.author_surname, book_name\r\n"
                + "FROM book  AS b INNER JOIN\r\n"
                + "(SELECT author_id, author_name, author_surname \r\n"
                + "FROM author WHERE author_name LIKE '" + name + "') \r\n"
                + "AS a ON b.author_id = a.author_id";
        String explainJoinText = "EXPLAIN ANALYZE (" + joinText + ")";
        try {
            Statement statement = connection.createStatement();
            ResultSet rs = statement.executeQuery(joinText);
            View.printTable(rs, 3);
            rs = statement.executeQuery(explainJoinText);
            View.printExplain(rs);
            statement.close();
            rs.close();
        }catch (SQLException e) {
            postgresError(e);
        }
    }

    // return number book, which reader take for a certain period
    public static void countBookInReader(String date) {
        String countBookInReader = "SELECT reader_name, reader_surname, count(*) AS ticket_count \r\n"
                + "FROM reader AS r INNER JOIN\r\n"
                + "(SELECT reader_id, date_issued FROM reader_ticket WHERE date_issued > '" + date + "')\r\n"
                + "AS rt ON r.reader_id = rt.reader_id\r\n"
                + "GROUP BY reader_name, reader_surname\r\n"
                + "ORDER BY ticket_count desc";
        String explainCountBookInReader = "EXPLAIN ANALYZE (" + countBookInReader + ")";
        try {
            Statement statement = connection.createStatement();
            ResultSet rs = statement.executeQuery(countBookInReader);
            View.printTable(rs, 3);
            rs = statement.executeQuery(explainCountBookInReader);
            View.printExplain(rs);
            statement.close();
            rs.close();
        }catch (SQLException e) {
            postgresError(e);
        }
    }
```

```java
        // return the most popular authors and their books
        public static void countPopularAuthor(String date) {
                String countPopularAuthor = "SELECT author_name, author_surname, count(*) AS ticket_count\r\n"
                                + "FROM author AS a INNER JOIN\r\n"
                                + "(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count\r\n"
                                + "FROM book AS b INNER JOIN\r\n"
                                + "(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')\r\n"
                                + "AS rt ON b.book_id = rt.book_id\r\n"
                                + "GROUP BY rt.book_id, b.book_name, author_id\r\n"
                                + "ORDER BY ticket_count desc)\r\n"
                                + "AS res ON res.author_id = a.author_id\r\n"
                                + "GROUP BY author_name, author_surname\r\n"
                                + "ORDER BY ticket_count desc";
                String explainCountPopularAuthor = "EXPLAIN ANALYZE (" + countPopularAuthor + ")";
                try {
                        Statement statement = connection.createStatement();
                        ResultSet rs = statement.executeQuery(countPopularAuthor);
                        View.printTable(rs, 3);
                        rs = statement.executeQuery(explainCountPopularAuthor);
                        View.printExplain(rs);
                        statement.close();
                        rs.close();
                }catch (SQLException e) {
                        postgresError(e);
                }
        }
```

```java
         // return the most popular authors and their books
         public static void countPopularAuthor(String date) {
                 String countPopularAuthor = "SELECT author_name, author_surname, count(*) AS ticket_count\r\n"
                                 + "FROM author AS a INNER JOIN\r\n"
                                 + "(SELECT rt.book_id, book_name, author_id, count(*) AS ticket_count\r\n"
                                 + "FROM book AS b INNER JOIN\r\n"
                                 + "(SELECT book_id, date_issued FROM reader_ticket WHERE date_issued > '2015-10-10')\r\n"
                                 + "AS rt ON b.book_id = rt.book_id\r\n"
                                 + "GROUP BY rt.book_id, b.book_name, author_id\r\n"
                                 + "ORDER BY ticket_count desc)\r\n"
                                 + "AS res ON res.author_id = a.author_id\r\n"
                                 + "GROUP BY author_name, author_surname\r\n"
                                 + "ORDER BY ticket_count desc";
                 String explainCountPopularAuthor = "EXPLAIN ANALYZE (" + countPopularAuthor + ")";
                 try {
                         Statement statement = connection.createStatement();
                         ResultSet rs = statement.executeQuery(countPopularAuthor);
                         View.printTable(rs, 3);
                         rs = statement.executeQuery(explainCountPopularAuthor);
                         View.printExplain(rs);
                         statement.close();
                         rs.close();
                 }catch (SQLException e) {
                         postgresError(e);
                 }
         }

         public static String[] getColumnsName(String table) {
                 String[] result = null;
                 try {
                         String getColumns = "SELECT column_name FROM information_schema.columns WHERE table_name = '" + table + "'";

                         Statement statement = connection.createStatement();
                         ResultSet rs = statement.executeQuery(getColumns);
                         int size = countColumns(table);
                         result = new String[size];

                         for(int i = 0; rs.next(); i++) {
                                 result[i] = rs.getString(1);
                         }
                         statement.close();
                         rs.close();
                 }catch (SQLException e) {
                         postgresError(e);
                         return null;
                 }

                 return result;
         }
```

```
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 countCoulumns = "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(countCoulumns);
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         public static void printTable(String table) {
359                 try {
360                         String printTable = "SELECT * FROM " + table;
361                         Statement statement = connection.createStatement();
362                         ResultSet rs = statement.executeQuery(printTable);
363                         int size = countColumns(table);
364                         View.printTable(rs, size);
365                         statement.close();
366                         rs.close();
367                 }catch (SQLException e) {
368                         postgresError(e);
369                 }
370         }
371
372         private static void postgresError(SQLException e) {
373                 System.out.println("\nError: Incorrect data entered");
374         }
375
376         public static void printColumnsNameType(String table) {
377                 String[] columnsName = getColumnsName(table);
378                 String[] columnsType = getColumnsType(table);
379
380                 int size = columnsName.length;
381
382                 String[][] columnsNameType = new String[size][2];
383
384                 for(int i = 0; i < size; i++) {
385                         columnsNameType[i][0] = columnsName[i];
386                         columnsNameType[i][1] = columnsType[i];
387                 }
388
389                 View.printMatrix(columnsNameType);
390         }
391 }
392
```

View.java

```java
package db.lab2.view;

import db.lab2.model.Model;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;



public class View {
	public static void printTable(ResultSet rs, int size) {
		try {
			while (rs.next()) {
				for(int i = 1; i <= size; i++) {
					System.out.printf("%-50s", rs.getString(i));
				}
				System.out.printf("\n");
			}
		} catch (Exception e) {
			System.out.println(e);
		}
	}

	public static void printExplain(ResultSet rs) {
		try {
			String time;
			while(rs.next()) {
				if((time = rs.getString(1)).startsWith("Execution Time:")) {
					System.out.println(time);
				}
			}
		}catch (SQLException e) {
		System.out.println(e);

		}
	}

	public static void performOperation(boolean bool) {
		if(bool) {
			System.out.println("\nOperation is performed");
		} else {
			System.out.println("\nOperation is not performed");
		}
	}

	public static int viewMenu() {
		String menu = "1. Add data\n"
					+ "2. Delete data\n"
					+ "3. Update data\n"
					+ "4. Input random data\n"
					+ "5. Search for books by author's name\n"
					+ "6. Search for readers with the largest number of books in a given period\n"
					+ "7. Search for the most popular authors for a certain period\n"
					+ "8. View table\n"
					+ "9. View table columns name\n"
					+ "10. Exit";
		Scanner in = new Scanner(System.in);
		System.out.println("\n" + menu);
		System.out.print("Enter the menu number: ");
		int numberMenu = in.nextInt();

		return numberMenu;
	}

	public static String[] inputAddData() {
		Scanner in = new Scanner(System.in);

		System.out.print("Input table: ");
		String table = in.nextLine();

		int numberColumns = Model.countColumns(table);
		String[] result = new String[numberColumns + 1];

		String[] columnsName = Model.getColumnsName(table);

		for(int i = 0; i < numberColumns; i++) {
			System.out.print("Input " + columnsName[i] + ": ");
			result[i] = in.nextLine();
		}

		result[numberColumns] = table;

		return result;
	}
```

```java
    public static String[] inputDeleteData() {
            Scanner in = new Scanner(System.in);

            String[] result = new String[3];

            System.out.print("Input table: ");
            result[0] = in.nextLine();

            System.out.print("Input the name of the column by which the data will be displayed(if delete all data in table input 'null'): ");
            result[1] = in.nextLine();

            System.out.print("Input value column: ");
            result[2] = in.nextLine();

            return result;
    }

     public static String[] inputUpdateData() {
             Scanner in = new Scanner(System.in);

             System.out.print("Input table: ");
             String table = in.nextLine();

             int numberColumns = Model.countColumns(table);
             String[] result = new String[numberColumns + 3];

             String[] columnsName = Model.getColumnsName(table);

             result[numberColumns] = table;

             System.out.print("Input the name of the column in which the data changes: ");
             result[numberColumns + 1] = in.nextLine();

             System.out.print("Input value column: ");
             result[numberColumns + 2] = in.nextLine();

             for(int i = 0; i < numberColumns; i++) {
                     System.out.print("Input " + columnsName[i] + ": ");
                     result[i] = in.nextLine();
             }

             return result;
     }
    public static String[] inputRandomData() {
            Scanner in = new Scanner(System.in);

            String[] result = new String[2];

            System.out.print("Input table( all tables, input 'all'): ");
            result[0] = in.nextLine();

            System.out.print("Input the amount of random data: ");
            result[1] = in.nextLine();

            return result;
    }

    public static String inputSearchByName() {
            Scanner in = new Scanner(System.in);

            System.out.print("Input name: ");
            String name = in.nextLine();

            return name;
    }

    public static String inputCountBookInReader() {
            Scanner in = new Scanner(System.in);

            System.out.print("Input date(YYYY-MM-DD): ");
            String date = in.nextLine();

            return date;
    }
```

```
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 numberRows = matrix.length;
182                 int numberColumns = matrix[0].length;
183 
184                 for(int i = 0; i < numberRows; i++) {
185                         for(int j = 0; j < numberColumns; j++) {
186                                 System.out.printf("%-20s", matrix[i][j]);
187                         }
188                         System.out.println();
189                 }
190         }
191 }
```