

- 1. Tools versions**
- 2. How to download project on your local environment (laptop, PC)**
- 3. How to setup database (Pgadmin 4, cmd, application.yml configuration)**
- 4. Liquibase contexts**
- 5. Additional information**

## 1. Tools versions

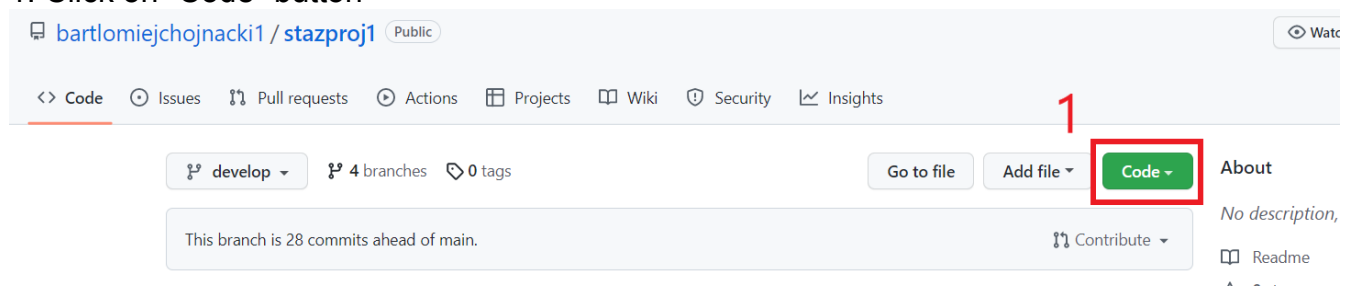
Java → 11  
PostgreSQL → 14.2  
Maven → 3.8.5  
Git → 2.35.1

## 2. How to download project on your local environment (laptop, PC)

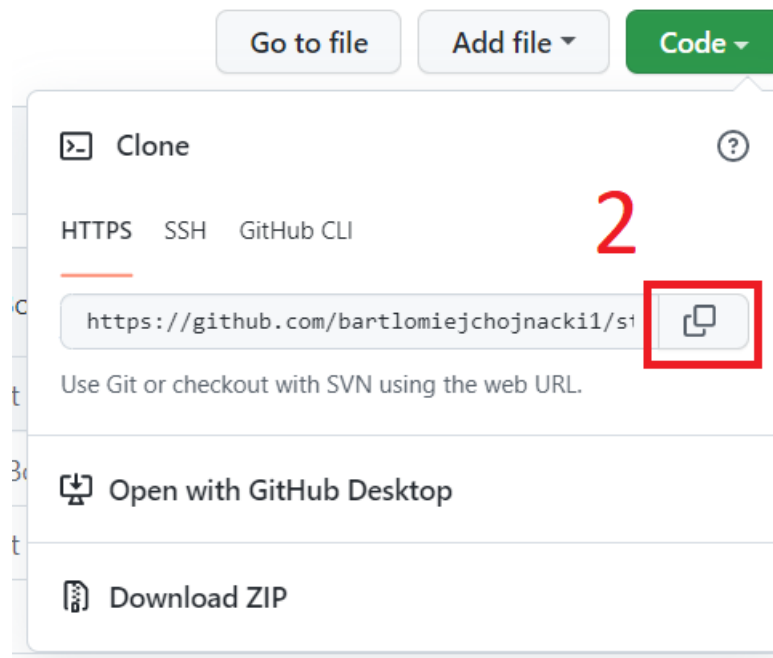
Repo link:

<https://github.com/bartlomiejchojnacki1/stazproj1>

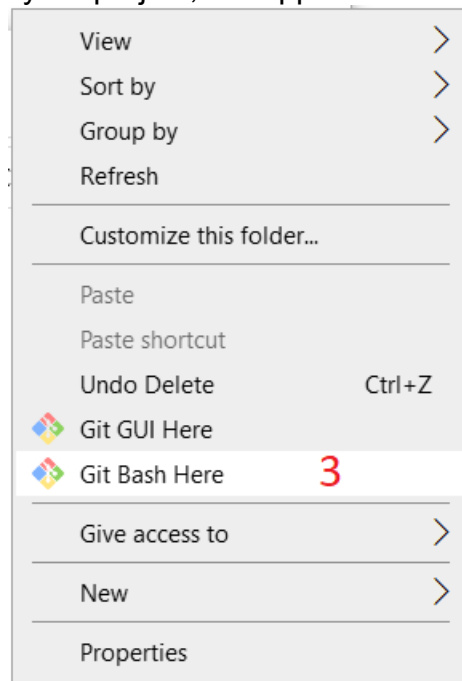
### 1. Click on “Code” button



### 2. Click on button to copy Https link to repo



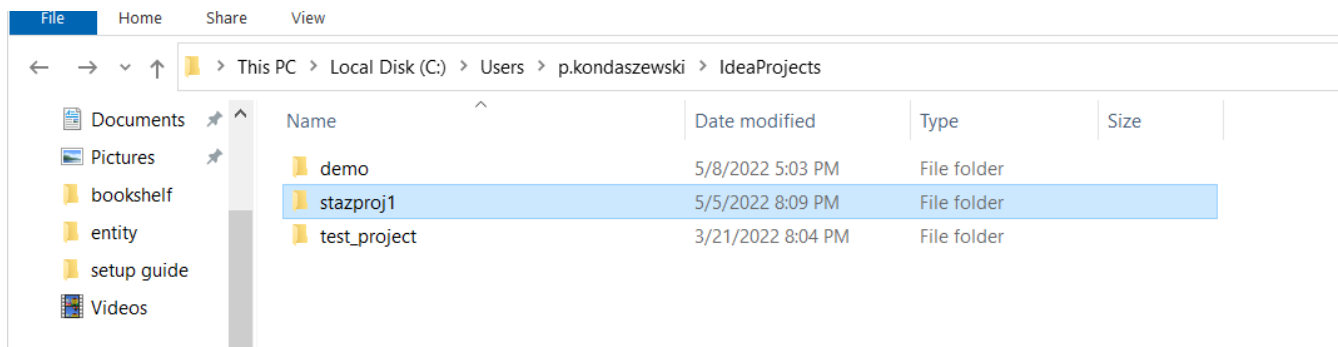
3. Go to folder where you want your project, click ppm and choose git bash option



4. Use git commit command

```
p.kondaszewski@OSZ1-LDL-P10153 MINGW64 ~/IdeaProjects
$ git clone https://github.com/bartlomiejchojnacki1/stazproj1.git
```

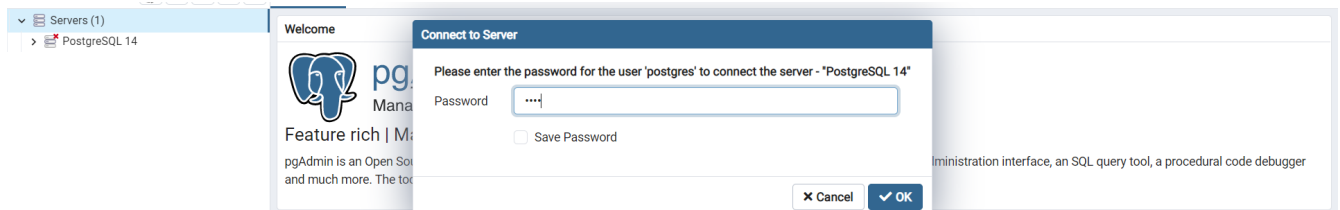
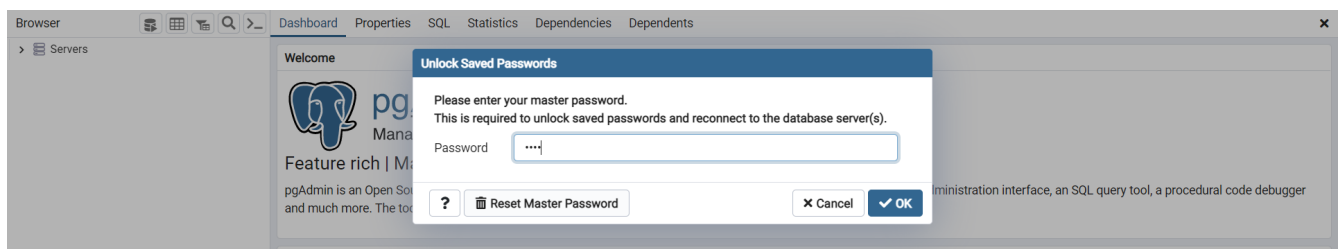
That's all. Project folder should now be accessible



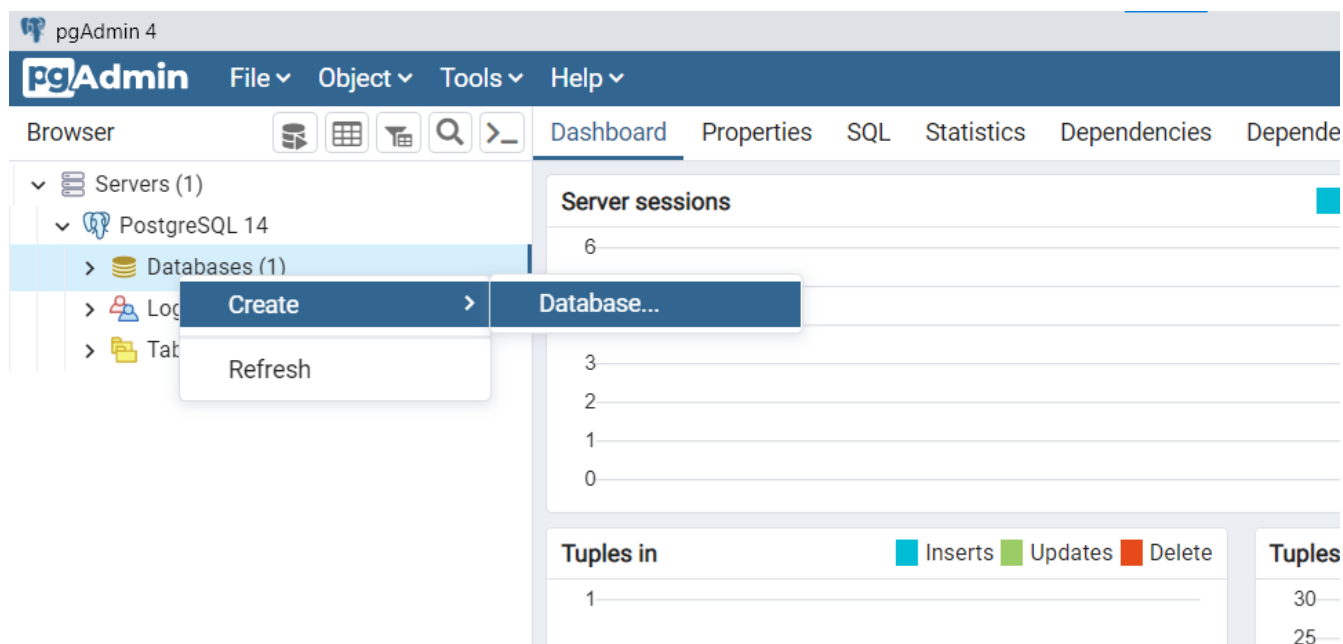
### 3. How to setup database (Pgadmin 4, cmd)

#### - Pgadmin 4

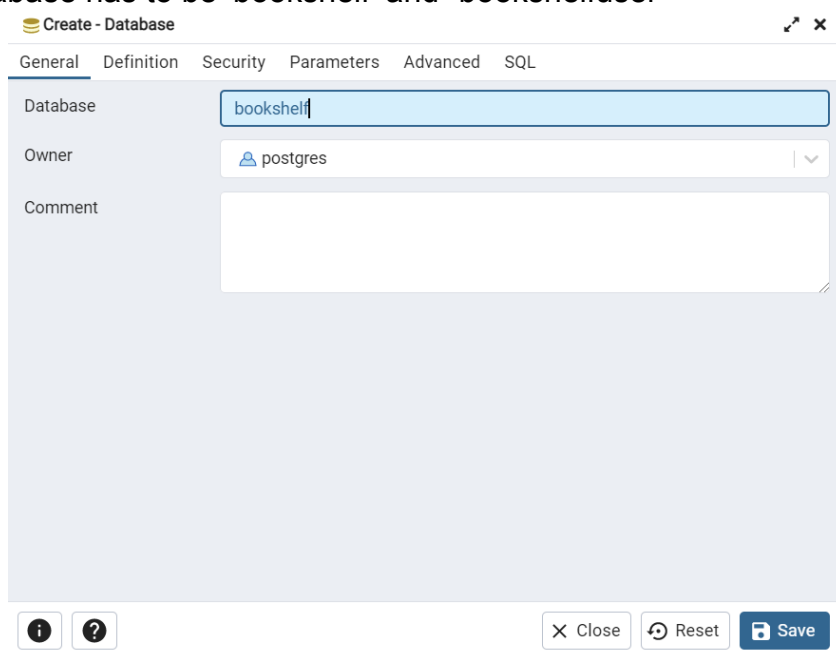
1. Log on PostgreSQL account and server with your passwords



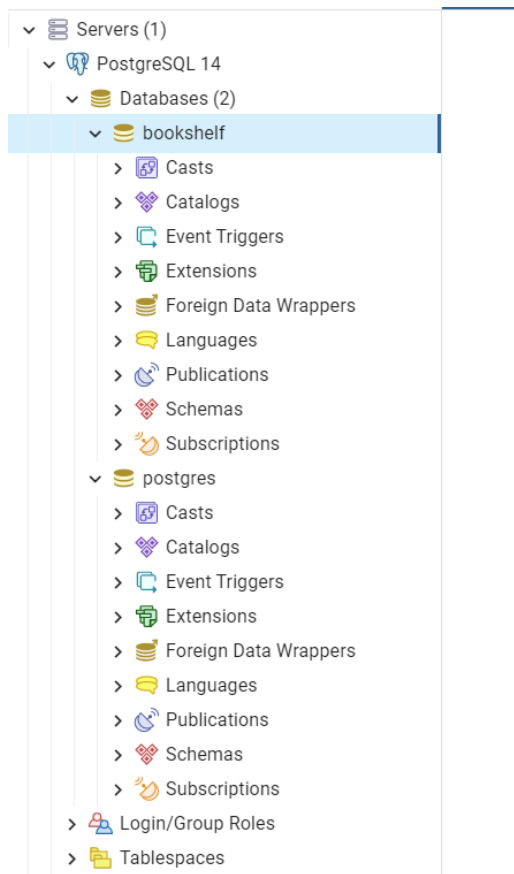
2. Create two new database with name “bookshelf” and “bookshelfuser”



Name of the database has to be 'bookshelf' and "bookshelfuser"

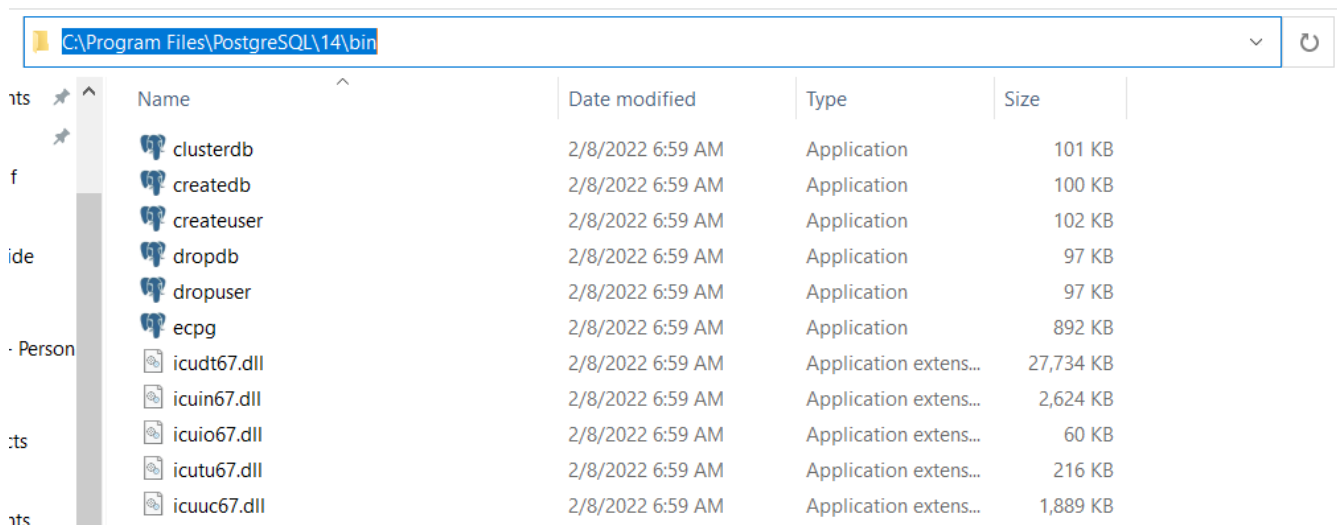


That's all from pgadmin. Database 'bookshelf' should looks like this.



## - cmd

1. Copy path to bin folder of PostgreSQL



2. Open cmd and go to bin folder

```
C:\Windows\System32>cd C:\Program Files\PostgreSQL\14\bin
C:\Program Files\PostgreSQL\14\bin>
```

3. Connect to psql with your PostgreSQL account with given command

```
C:\Program Files\PostgreSQL\14\bin>psql -U postgres
Password for user postgres:
psql (14.2)
WARNING: Console code page (852) differs from Windows code page (1250)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

Where 'postgres' is the username of the account. After that you have to write a password.

4. Use 'CREATE DATABASE' command

Name of the database has to be 'bookshelf'. After that, just for verification of 'bookshelf' creation, use meta-command '\list' or shortcut '\l'.

```
postgres=# CREATE DATABASE bookshelf;
CREATE DATABASE
postgres=#
postgres=# \l
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
bookshelf	postgres	UTF8	English_United States.1250	English_United States.1250	
postgres	postgres	UTF8	English_United States.1250	English_United States.1250	
template0	postgres	UTF8	English_United States.1250	English_United States.1250	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	English_United States.1250	English_United States.1250	=c/postgres + postgres=CTc/postgres

```
(4 rows)
```

After that your database is ready to go.

## - application.yml configuration

To run database in application correctly, you have to change username, password in application.yml file

(remove username, password and write your own personal data where red underlines are)

```
server:
  port : 8467
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/bookshelf
    username: postgres
    password: root
    # password: admin
  jpa:
    hibernate:
      ddl-auto: update
      show-sql: true
      database: postgresql
      database-platform: org.hibernate.dialect.PostgreSQLDialect
    liquibase:
      contexts: prod
      # contexts: test
      dropFirst: false
```

```
server:
  port : 8466
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/bookshelfuser
    username: postgres
    #password: root
    password: admin
  jpa:
    hibernate:
      ddl-auto: update
      show-sql: true
      database: postgresql
      database-platform: org.hibernate.dialect.PostgreSQLDialect
    liquibase:
      #contexts: prod
      contexts: test
      dropFirst: false
```

## 4. Liquibase contexts

The application provides two database contexts



```

server:
  port : 8467
spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/bookshelf
    username: postgres
    password: root
    # password: admin
  jpa:
    hibernate:
      ddl-auto: update
      show-sql: true
      database: postgresql
      database-platform: org.hibernate.dialect.PostgreSQLDialect
    liquibase:
      contexts: prod
      # contexts: test
      dropFirst: false

```

When you want to your database without any data, just setup of the entities, use **prod** contexts.

On the other hand, when you want your database to start with defined data in it, use **test** context.

```

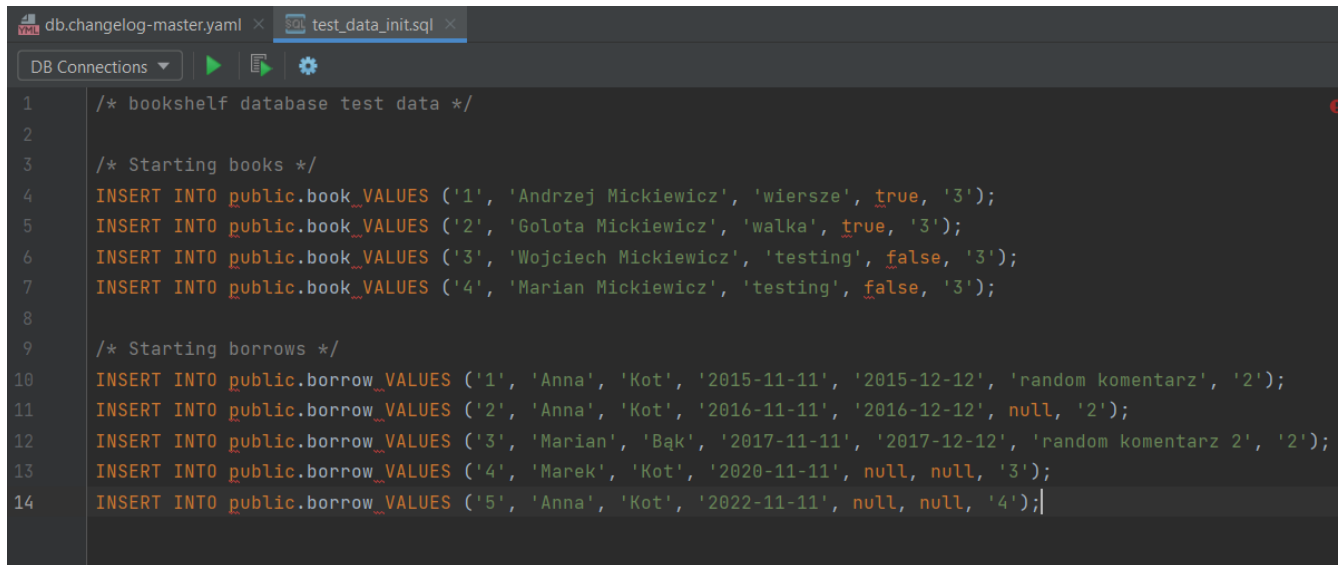
db.changelog-master.yaml
1  databaseChangeLog:
2    - changeSet:
3      id: create_book_table
4      author: gl
5      context: prod, test
6      changes: <3 items>
96   - include:
97     - context: prod, test
98     - file: /db/changelog/changes/category_init.sql
99   - include:
100     - context: test
101     - file: /db/changelog/changes/test_data_init.sql

```

1. Structure of the database (entities, foreign keys etc.)

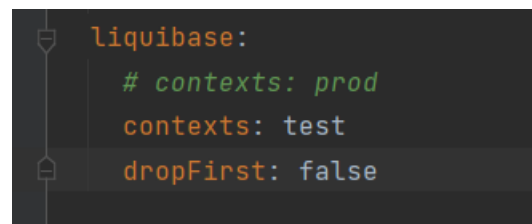
2. Starting categories (has to be in both contexts)
3. Test data (only in test context)

You can edit your test data in changes/test\_data\_init.sql file



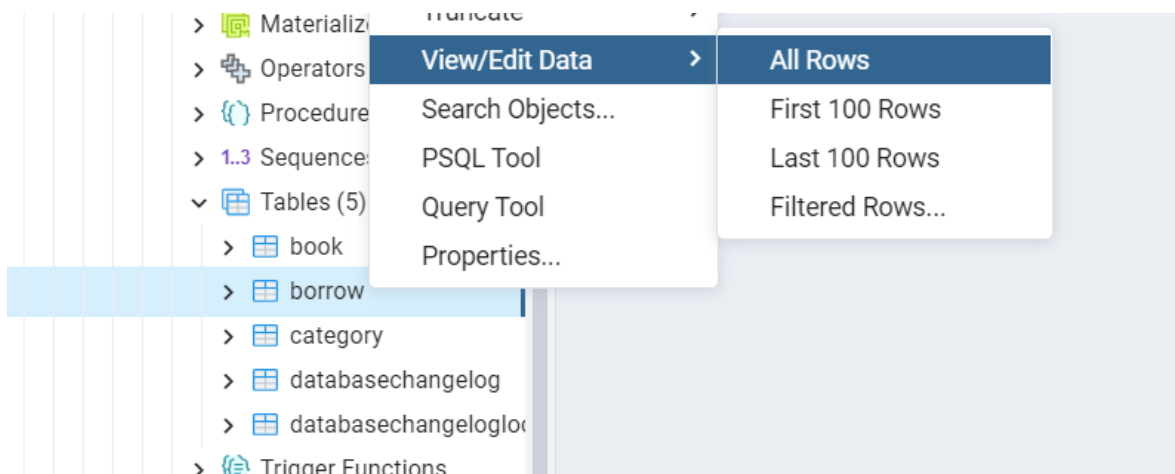
```
db.changelog-master.yaml x test_data_init.sql x
DB Connections
1  /* bookshelf database test data */
2
3  /* Starting books */
4  INSERT INTO public.book__VALUES ('1', 'Andrzej Mickiewicz', 'wiersze', true, '3');
5  INSERT INTO public.book__VALUES ('2', 'Golota Mickiewicz', 'walka', true, '3');
6  INSERT INTO public.book__VALUES ('3', 'Wojciech Mickiewicz', 'testing', false, '3');
7  INSERT INTO public.book__VALUES ('4', 'Marian Mickiewicz', 'testing', false, '3');
8
9  /* Starting borrows */
10 INSERT INTO public.borrow__VALUES ('1', 'Anna', 'Kot', '2015-11-11', '2015-12-12', 'random komentarz', '2');
11 INSERT INTO public.borrow__VALUES ('2', 'Anna', 'Kot', '2016-11-11', '2016-12-12', null, '2');
12 INSERT INTO public.borrow__VALUES ('3', 'Marian', 'Bak', '2017-11-11', '2017-12-12', 'random komentarz 2', '2');
13 INSERT INTO public.borrow__VALUES ('4', 'Marek', 'Kot', '2020-11-11', null, null, '3');
14 INSERT INTO public.borrow__VALUES ('5', 'Anna', 'Kot', '2022-11-11', null, null, '4');
```

To verify if test context works, change context of the application in application.yml to contexts: test











```
liquibase:
  # contexts: prod
  contexts: test
  dropFirst: false
```

After that search for inserted data in your database (pgadmin 4).



There are test data, now you are able to data processing

	 id [PK] integer 	firstname character varying (25) 	surname character varying (25) 	borrowed date 	returned date 	comment character varying (255) 	book_id integer 
1	1	Anna	Kot	2015-11-11	2015-12-12	random komentarz	2
2	2	Anna	Kot	2016-11-11	2016-12-12	[null]	2
3	3	Marian	Bak	2017-11-11	2017-12-12	random komentarz 2	2
4	4	Marek	Kot	2020-11-11	[null]	[null]	3
5	5	Anna	Kot	2022-11-11	[null]	[null]	4

## 5. Additional information

- Code coverage level = 75% and above
- Use JavaDocs to describe classes and methods
- Meaningful commits descriptions (**use** git squash, **avoid** git merge)