

KURS SQL

GIT

A. Wprowadzenie

Poniżej znajdziesz najważniejsze polecenia git oraz przydatne odnośniki.

- **State approach** – podejście zakładające, że przechowujemy definicję wszystkich obiektów (CREATE TABLE/ VIEW/ PROCEDURE itd.). Podczas wdrażania zmian – generowany jest skrypt różnicowy – na podstawie porównania tych skryptów i bazy danych (Visual Studio/ Azure Data Studio)
- **Migration approach** – podejście zakładające, że przechowujemy skrypty zmian – kolejnych wersji (trzeba je wykonywać w określonej kolejności)

1. Odnośniki

- Instalacja Git
<https://git-scm.com/>
- Git Ściąga
<https://training.github.com/downloads/pl/github-git-cheat-sheet/>
- GitHub
<https://github.com/>
- GitHub Desktop
<https://desktop.github.com/>

2. Najważniejsze polecenia

\$ git --version	Pokazuje wersję zainstalowanego git
\$ git config --global user.name	Ustawia nazwę użytkownika (dołączoną do commitów)
\$ git config --global user.email	Ustawia adres e-mail użytkownika
\$ git init [nazwa]	Utworzenie nowego lokalnego repozytorium
rm -rf .git	Usuwanie repozytorium (katalogu .git), pozostawiając pliki
\$ git clone [url]	Kopiowanie istniejącego repo i jego historię
\$ git status	Sprawdzenie statusu lokalnych plików (do zatwierdzenia)
\$ git diff	Zmiany wprowadzone do projektu (jeszcze niezatwierdzone)
\$ git add [plik]	Dodaje pliki do Stage
\$ git diff --staged	Pokazuje różnice między plikami w stage, a ostatnim commit-em
\$ git commit	Zatwierdzenie plików/ migawki zmian
\$ git push	Wysyła zmiany (zacomitowane) do zdalnego repo
\$ git pull	Pobiera zmiany innych członków zespołu ze zdalnego repo
\$ git log	Wyświetla log zmian
\$ git show [commitid]	Wyświetlenie zawartości commita

B. Instalacja Git

1. Przejdź pod adres <https://git-scm.com/>
2. Pobierz i zainstaluj najświeższą wersję git
 - Zastosuj domyślne ustawienia
 - W ostatnim kroku instalatora zaznacz **Launch GitBash**
3. Upewnij się, że git został zainstalowany - poprzez sprawdzenie wersji: `git --version`
4. Ustaw imię i nazwisko i adres e-mail – podaj swoje własne dane. Ustawienia te będą stosowane do wszystkich nowych repozytoriów

```
git config --global user.name "TWOJE IMIE I NAZWISKO"
```

```
git config --global user.email "TWÓJ ADRES"
```



```
MINGW64:/c/Users/kowalski

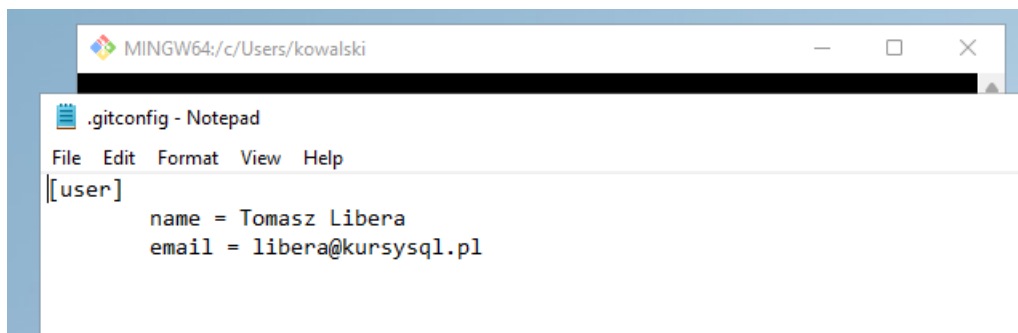
kowalski@amida1a MINGW64 ~
$ git --version
git version 2.37.3.windows.1

kowalski@amida1a MINGW64 ~
$ git config --global user.name "Tomasz Libera"

kowalski@amida1a MINGW64 ~
$ git config --global user.email "libera@kursysql.pl"

kowalski@amida1a MINGW64 ~
$
```

5. Możesz sprawdzić plik z konfiguracją, otwierając go w notatniku: C:\Users\TWÓJ-USER\.gitconfig



6. Zamknij konsolę Bash

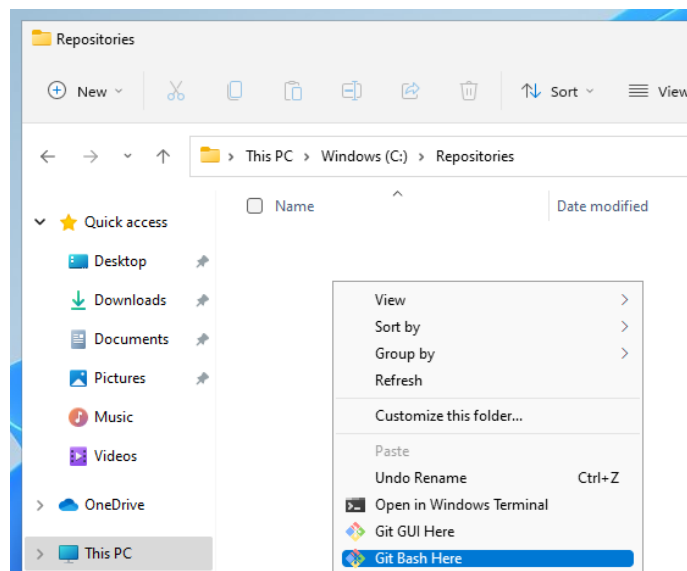
C. Tworzenie repozytorium

W ramach tego ćwiczenia nauczysz się tworzyć nowe repozytorium, wykonywać pierwszy commit, a także klonować istniejące repo z github.

1. Tworzenie nowego repozytorium

- Utwórz folder **C:\Repositories**
- Utwórz podfolder **myfirstrepo**
- Otwórz konsolę Bash
Możesz również używać standardowej konsoli Windows
- Utwórz nowe repozytorium o nazwie **myfirstrepo**, używając poniższego polecenia:

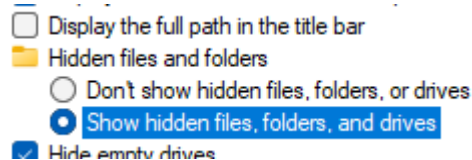
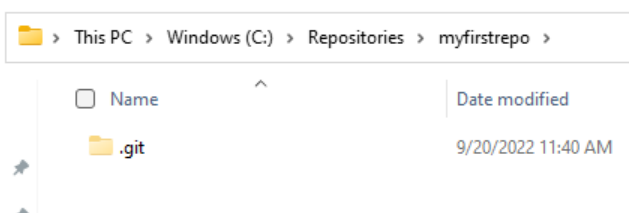
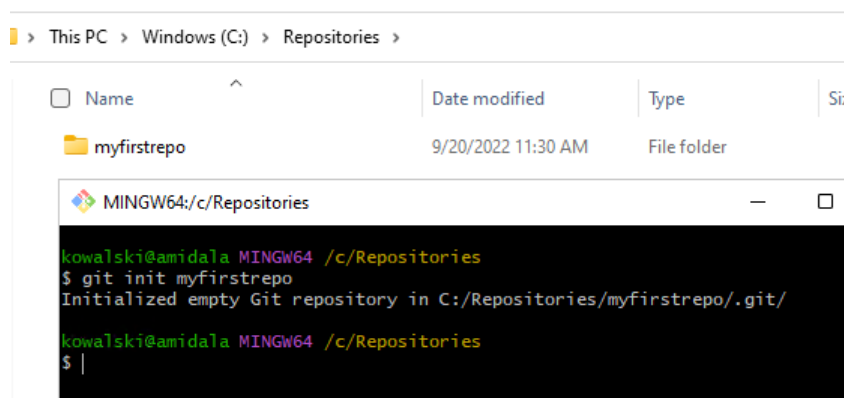
```
git init myfirstrepo
```



- Spowoduje to utworzenie repozytorium w nowoutworzonym folderze.

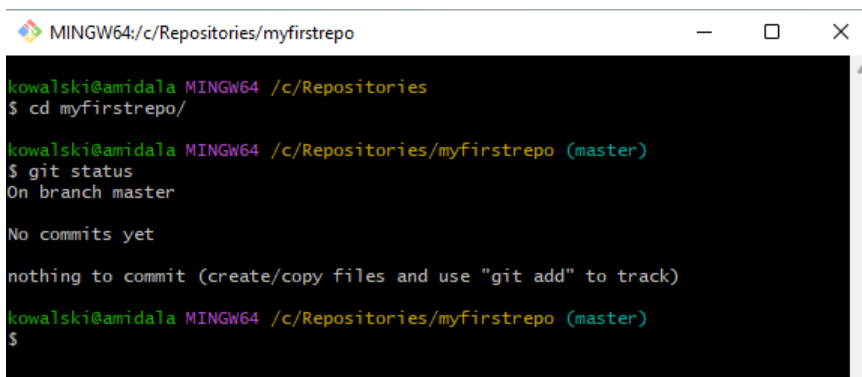
Analogicznie można utworzyć folder i wywołać w nim polecenie **git init** (bez wskazywania nazwy) – wówczas repozytorium będzie utworzone w bieżącym folderze.

- Sprawdź zawartość folderu **myfirstrepo** – wydaje się być pusty, ale jeśli włączyć wyświetlanie folderów ukrytych – odnajdziesz tam



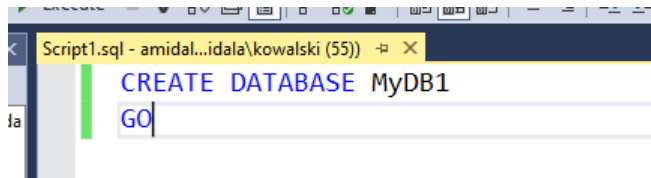
folder **.git** – zawierający bazę danych git

- W przypadku chęci skasowania bazy repozytorium – bez kasowania plików wchodzących w jego skład – wystarczy skasować ten folder. Nie rób tego teraz :)
- Korzystając z konsoli Bash, przejdź do folderu **myfirstrepo** i wyświetl status – otrzymasz komunikat o braku commitów i braku plików do „zakomitownia”.



2. Dodanie pierwszych plików do repo

- a. Korzystając z SQL Server Management Studio albo Azure Data Studio – utwórz plik Script1.sql, zawierający poniższy kod i zapisz go w folderze myfirstrepo



- b. Ponownie wykonaj `git status`

Zostanie wyświetlony komunikat, że folder zawiera pliki (plik), które nie są śledzone i należy je dodać do repo

```
kowalski@amida MINGW64 /c/Repositories/myfirstrepo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Script1.sql

nothing added to commit but untracked files present (use "git add" to track)
```

- c. Wykonaj `git add Script1.sql`

Możesz skorzystać z podpowiadania nazw plików – naciskając Tab po wpisaniu pierwszych liter. Po dodaniu pliku wykonaj `git status` aby ponownie sprawdzić stan plików w folderze.

```
kowalski@amida MINGW64 /c/Repositories/myfirstrepo (master)
$ git add Script1.sql

kowalski@amida MINGW64 /c/Repositories/myfirstrepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Script1.sql
```

- d. W ramach ćwiczenia dodaj 2-3 pliki o dowolnych nazwach i również dodaj je, możesz wykonać kilkakrotnie polecenie `add`, albo dodać je w ramach jednego polecenia:

`git add plik1 plik2 plik3 itd`

3. Pierwszy commit

*Dodane pliki czekają na zatwierdzenie poleceniem **git commit** – tworzone jest wówczas coś na wzór migawki projektu, albo jej kopii zapasowej. Każdy commit zawiera nazwę autora oraz krótki opis. Kolejne commit-y reprezentują kolejne wersje projektu.*

Przed commit-em należy sprawdzić czy wszystkie wymagane pliki zostały dodane (`git status`)

```
kowalski@amida MINGW64 /c/Repositories/myfirstrepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   GetDate.sql
        new file:   Script1.sql
        new file:   Tab1.sql
```

- a. Wywołaj `git commit`.

Spowoduje to otwarcie domyślnego edytora (ustawionego podczas instalacji – aktualnie Vim).

Tekst który się w nim znajduje jest oznaczony jako komentarz i jest to:

- Informacja o na jakiej gałęzi (branch) się znajdujemy
- Informacja, że to początkowy – pierwszy commit tego repo
- Lista plików – dodanych

Aby dodać opis commita, należy nacisnąć na klawiaturze klawisz **A** i wpisać opis (np.: „**Add first 3 files to project**”), następnie zapisać wybierając kolejno na klawiaturze (specyficzne dla edytora Vim):

- Esc
- :wq
- Enter

```
MINGW64:/c/Repositories/myfirstrepo
Add 3 first files to project
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   GetDate.sql
#   new file:   Script1.sql
#   new file:   Tab1.sql
```

- b. Zmiany zostały zatwierdzone, co będzie podsumowane odpowiednim komunikatem

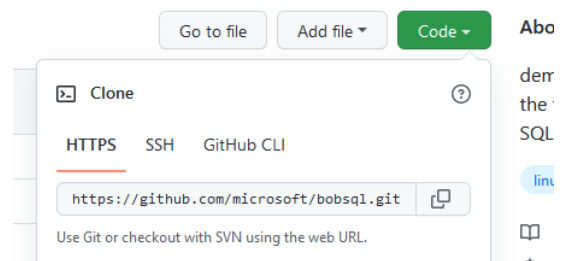
```
kowalski@amida1a MINGW64 /c/Repositories/myfirstrepo (master)
$ git commit
[master (root-commit) 1849f94] Add 3 first files to project
3 files changed, 4 insertions(+)
create mode 100644 GetDate.sql
create mode 100644 Script1.sql
create mode 100644 Tab1.sql
```

- c. W kolejnej lekcji dowiesz się jak przeglądać i porównywać poszczególne zmiany w projekcie.

4. Klonowanie repozytorium

Poza tworzeniem nowego repozytorium, istnieje też możliwość pobrania istniejącego. W tym ćwiczeniu skorzystamy z dostępnego publicznie repozytorium zawierającego (świetne) materiały dot. SQL Server autorstwa Bob Ward.

- a. Korzystając z przeglądarki, przejdź pod adres <https://github.com/microsoft/bobsq1>
- b. Wybierz przycisk **Code** i skopiuj adres url repo
- c. W ramach Bash git-a przejdź do folderu głównego na Twoje repozytoria (C:\Repositories) i wykonaj: `git clone [url]`
- d. Poczekaj na skopiowanie wszystkich plików lokalnie



```
kowalski@amida1a MINGW64 /c/Repositories/myfirstrepo (master)
$ cd C:\Repositories

kowalski@amida1a MINGW64 /c/Repositories
$ git clone https://github.com/microsoft/bobsq1.git
Cloning into 'bobsq1'...
remote: Enumerating objects: 3411, done.
remote: Counting objects: 100% (727/727), done.
remote: Compressing objects: 100% (284/284), done.
remote: Total 3411 (delta 421), reused 722 (delta 417), pack-reused 2684
Receiving objects: 100% (3411/3411), 177.18 MiB | 22.29 MiB/s, done.
Resolving deltas: 100% (1651/1651), done.
Updating files: 100% (10634/10634), done.
```

- e. W celu sprawdzenia, czy Bob Ward nie udostępnił nowych materiałów w swoim repo – przejdź do folderu **sqlbobs** (folderu repozytorium) i wykonaj `git pull [repo-url]`

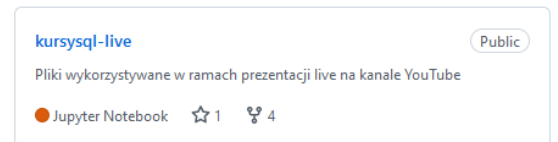
```
kowalski@amida1a MINGW64 /c/Repositories
$ cd bobsq1/

kowalski@amida1a MINGW64 /c/Repositories/bobsq1 (master)
$ git pull https://github.com/microsoft/bobsq1.git
From https://github.com/microsoft/bobsq1
* branch      HEAD      -> FETCH_HEAD
Already up to date.
```

D. Podstawowe polecenia

W tym ćwiczeniu poznasz podstawowe polecenia pozwalające na poruszanie się po repozytorium zawierającego pliki projektu bazdanowego (State approach).

Poznasz takie polecenia jak: *add, log, show, diff, checkout, revert*



1. Pobierz pliki prostego projektu bazdanowego Filmy

- Przejdź pod adres <https://github.com/kursysql/>
- Otwórz repozytorium, zawierające pliki webinarów.
- Przejdź do folderu **webinar-20220920-GIT**
- Pobierz i rozpakuj plik Filmy.zip do folderu **C:\Repositories**
- Otwórz git bash w C:\Repositories\Filmy i zainicjalizuj nowe repo:

```
git init
```

- Dodaj wszystkie istniejące pliki:

```
git add *
```

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Data/DaneTestowe.sql
        new file:   Procedures/DodajAktora.sql
        new file:   Procedures/DodajFilm.sql
```

- Wykonaj commit z opisem: **Database Filmy initial**

2. Poprawki typów danych

- Zapoznaj się z definicją tabel, korzystając z SSMS lub ADS.
- Popraw typy danych z char na nvarchar w tabelach:
 - dbo.Osoba
 - dbo.Gatunek
 - dbo.FilmAktor
 - dbo.Film
- Dodaj wszystkie zmienione pliki tabel do Stage

```
git add Tables/*
```

- Wykonaj comit z opisem **Correction of data types**

```
git commit
```

```
Correction of data types
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   Tables/Film.sql
#   modified:   Tables/FilmAktor.sql
#   modified:   Tables/Gatunek.sql
#   modified:   Tables/Osoba.sql
#
```

3. Dodanie obsługi błędów i transakcji do 2- 3 procedur składowanych

- Dokonaj zmian w procedurach składowanych – dodaj obsługę błędów i transakcji
- Dodaj wszystkie zmiany do Stage
- Wykonaj commit: **Add transactions to procs**

4. Dodanie nowego widoku

- a. Utwórz skrypt **Views\Aktorzy.sql** widoku zgodnie z poniższą definicją

```
CREATE VIEW Aktorzy
AS
SELECT Aktor.*, Osoba.Imie, Osoba.Nazwisko
FROM Osoba
JOIN Aktor ON Osoba.OsobaID = Aktor.OsobaID
```

- b. Dodaj nowy plik do Stage

```
git add Vews/Aktorzy.sql
```

- c. Wykonaj commit: **Add new view**

Tym razem posłuż się poniższym poleceniem;

```
git commit -m „Add new view”
```

5. Sprawdzanie historii

- a. Użyj poniższego polecenia aby sprawdzić historię zmian w projekcie:

```
git log
```

- b. W celu odwrócenia porządku sortowania – od najstarszych commit-ów użyj reverse:

```
git log --reverse
```

- c. Ograniczenie liczby zwróconych commit-ów:

```
git log -2
```

- d. Wyświetlenie commit-ów utworzonych przez konkretną osobę:

```
git log -author="Tomasz Libera"
```

- e. Wyświetl zawartość commita (użyj pierwszych 7 znaków identyfikatora commit-a który Cię interesuje)

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git log
commit 27e475e7c42062a6b5c08749f883493c95d9e6bf (HEAD -> m
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:54:02 2022 +0000

    Add new view

commit 8a4e87cb4316b90b31d02adafc5da3e7f6f29142
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:46:36 2022 +0000

    Add transactions to procs

commit 9796d8563ac0ca1369299dbf0b00e71d2e4a6c66
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:40:03 2022 +0000

    Correction of data types

commit 7f293e7c9e8d97dbc145170754435280174d5b21
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:23:50 2022 +0000

    Database Filmy Initial

kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ |
```

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git show 8a4e87c
commit 8a4e87cb4316b90b31d02adafc5da3e7f6f29142
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:46:36 2022 +0000

    Add transactions to procs

diff --git a/Procedures/DodajRezysera.sql b/Procedures/DodajRezysera.sql
index 7ab867b..0baf554 100644
--- a/Procedures/DodajRezysera.sql
+++ b/Procedures/DodajRezysera.sql
@@ -19,6 +19,7 @@ BEGIN
    END

    BEGIN TRY -- !!!
+   BEGIN TRAN

        IF EXISTS(SELECT * FROM Osoba WHERE Nazwisko = @Nazwisko AND Imie = @Imie)
        BEGIN
@@ -44,10 +45,12 @@ BEGIN
        SELECT SCOPE_IDENTITY() AS RezyserID

    END

+   COMMIT
+   -- !!! start
    END TRY
    BEGIN CATCH
```


- f. Sprawdź czy pomiędzy ostatnim commit-em, a Twoją wersją roboczą są jakieś różnice:

```
git diff
```

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git diff
```

- g. Dokonaj modyfikacji jednego z obiektów; np. usuń opcjonalne nawiasy kwadratowe w definicji tabeli dbo.Kraj i ponownie wykonaj git diff

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git diff
diff --git a/Tables/Aktor.sql b/Tables/Aktor.sql
index 6040fd3..ab177a1 100644
--- a/Tables/Aktor.sql
+++ b/Tables/Aktor.sql
@@ -1,8 +1,8 @@
-CREATE TABLE [dbo].[Aktor] (
-  [AktorID] INT IDENTITY (1, 1) NOT NULL,
-  [OsobaID] INT NOT NULL,
-  PRIMARY KEY CLUSTERED ([AktorID] ASC),
-  CONSTRAINT [FK_Aktorzy_Osoby] FOREIGN KEY ([OsobaID]) REFERENCES [dbo].[Osoba] ([OsobaID])
+CREATE TABLE dbo.Aktor (
+  AktorID INT IDENTITY (1, 1) NOT NULL,
+  OsobaID INT NOT NULL,
```

- h. W celu sprawdzenia różnic tylko w jednym pliku, użyj

```
git diff Tables/Kraj.sql
```

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git diff Tables/Kraj.sql
diff --git a/Tables/Kraj.sql b/Tables/Kraj.sql
index b9f07ad..e360dad 100644
--- a/Tables/Kraj.sql
+++ b/Tables/Kraj.sql
@@ -1,9 +1,7 @@
-CREATE TABLE [dbo].[Kraj] (
-  [KrajID] INT IDENTITY (1, 1) NOT NULL,
-  [Kraj] NVARCHAR (100) NOT NULL,
-  PRIMARY KEY CLUSTERED ([KrajID] ASC)
+CREATE TABLE dbo.Kraj (
+  KrajID INT IDENTITY (1, 1) NOT NULL,
+  Kraj NVARCHAR (100) NOT NULL,
+  PRIMARY KEY CLUSTERED (KrajID ASC)
+);
-
GO
```

- i. Przed wykonaniem commit, warto sprawdzać różnice między plikami w Stage i w commit:

```
git --staged
```

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git diff --staged

kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tables/Aktor.sql
        modified:   Tables/Kraj.sql

no changes added to commit (use "git add" and/or "git commit -a")

kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git add *

kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git diff --staged
diff --git a/Tables/Aktor.sql b/Tables/Aktor.sql
index 6040fd3..ab177a1 100644
--- a/Tables/Aktor.sql
+++ b/Tables/Aktor.sql
@@ -1,8 +1,8 @@
-CREATE TABLE [dbo].[Aktor] (
-  [AktorID] INT IDENTITY (1, 1) NOT NULL,
-  [OsobaID] INT NOT NULL,
```


6. Przełączanie się pomiędzy wersjami

- a. Utwórz commit, zatwierdzając ostatnie zmiany *Cleanup table definitions*

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Tables/Aktor.sql
        modified:   Tables/Kraj.sql

kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git commit -m "Cleanup tables definitions"
[master ae27b60] Cleanup tables definitions
 2 files changed, 9 insertions(+), 11 deletions(-)
```

- b. Sprawdź jeszcze raz historię commit-ów

`git log`

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git log
commit ae27b60540f7825aada8db0021333d80c5853bd (HEAD -> master)
Author: Tomasz Libera <libera@kursysql.pl>
Date:   Tue Sep 20 14:25:06 2022 +0000

    Cleanup tables definitions

commit 27e475e7c42062a6b5c08749f883493c95d9e6bf
Author: Tomasz Libera <libera@kursysql.pl>
Date:   Tue Sep 20 13:54:02 2022 +0000

    Add new view
```

- c. Aby pobrać z bazy poprzednią wersję projektu wystarczy wykonać polecenie checkout, wskazując identyfikator (pierwszych 7 znaków) commita do którego wersji chcemy się odwołać.
W celu wycofania ostatnich zmian w definicji tabel – wskażę identyfikator **Add new view**

`git checkout 27e475e`

```
kowalski@amida1a MINGW64 /c/Repositories/Filmy (master)
$ git checkout 27e475e
Note: switching to '27e475e'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 27e475e Add new view
```

- d. Sprawdź definicję tabel **dbo.Kraj** lub **dbo.Aktor** aby przekonać się, że poprawki zostały wycofane

```
Kraj.sql - amida1a.F...ida1a\kowalski (56) -p x
CREATE TABLE [dbo].[Kraj] (
    [KrajID] INT IDENTITY (1, 1) NOT NULL,
    [Kraj] NVARCHAR (100) NOT NULL,
    PRIMARY KEY CLUSTERED ([KrajID] ASC)
```

- e. Możesz w każdym momencie wrócić do najnowszego commit, najpierw jednak wycofać się do początkowego commit-a. Najpierw musimy ustalić jego identyfikator:

```
kowalski@amida MINGW64 /c/Repositories/Filmy ((27e475e...))
$ git log --reverse
commit 7f293e7c9e8d97dbc145170754435280174d5b21
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:23:50 2022 +0000

    Database Filmy Initial
```

- f. ...w celu wykonania checkout

```
kowalski@amida MINGW64 /c/Repositories/Filmy ((27e475e...))
$ git checkout 7f293e7c
Previous HEAD position was 27e475e Add new view
HEAD is now at 7f293e7 Database Filmy Initial
```

- g. Teraz w procedurach znowu nie ma obsługi transakcji, brakuje też definicji widoku.
h. Wykonaj poniższe polecenie aby wrócić do najświeższej wersji głównej gałęzi master

git checkout master

```
kowalski@amida MINGW64 /c/Repositories/Filmy ((7f293e7...))
$ git checkout master
Previous HEAD position was 7f293e7 Database Filmy Initial
Switched to branch 'master'
```

- i. Następnie sprawdź czy historia uwzględni wszystkie commit-y i czy plik z definicją widoku jest dostępny

```
kowalski@amida MINGW64 /c/Repositories/Filmy (master)
$ git log
commit ae27b60540f7825aaeda8db0021333d80c5853bd (HEAD -> master)
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 14:25:06 2022 +0000

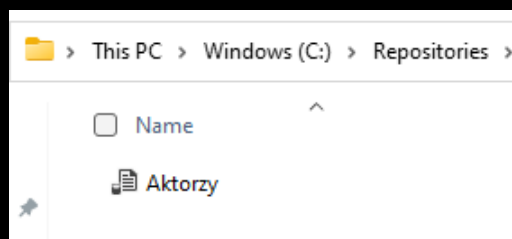
    Cleanup tables definitions

commit 27e475e7c42062a6b5c08749f883493c95d9e6bf
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:54:02 2022 +0000

    Add new view

commit 8a4e87cb4316b90b31d02adafc5da3e7f6f29142
Author: Tomasz Libera <libera@kursysql.pl>
Date: Tue Sep 20 13:13:15 2022 +0000

    Add new view
```



7. Wycofanie commit-a

- a. Założmy, że ostatnia zmiana polegająca na usunięciu (opcjonalnych) nawiasów kwadratowych nie spotkała się z aprobatą i chcemy ją wycofać.
Należy sprawdzić identyfikator commit-a do wycofania i użyć go w poleceniu revert.
W praktyce git revert utworzy nowy commit, wycofujący zmiany. Pojawi się więc opis modyfikacji, który w tym przypadku można po prostu zatwierdzić (Esc, :wq)

git revert ae27b60

```
Revert "Cleanup tables definitions"

This reverts commit ae27b60540f7825aaeda8db0021333d80c5853bd.

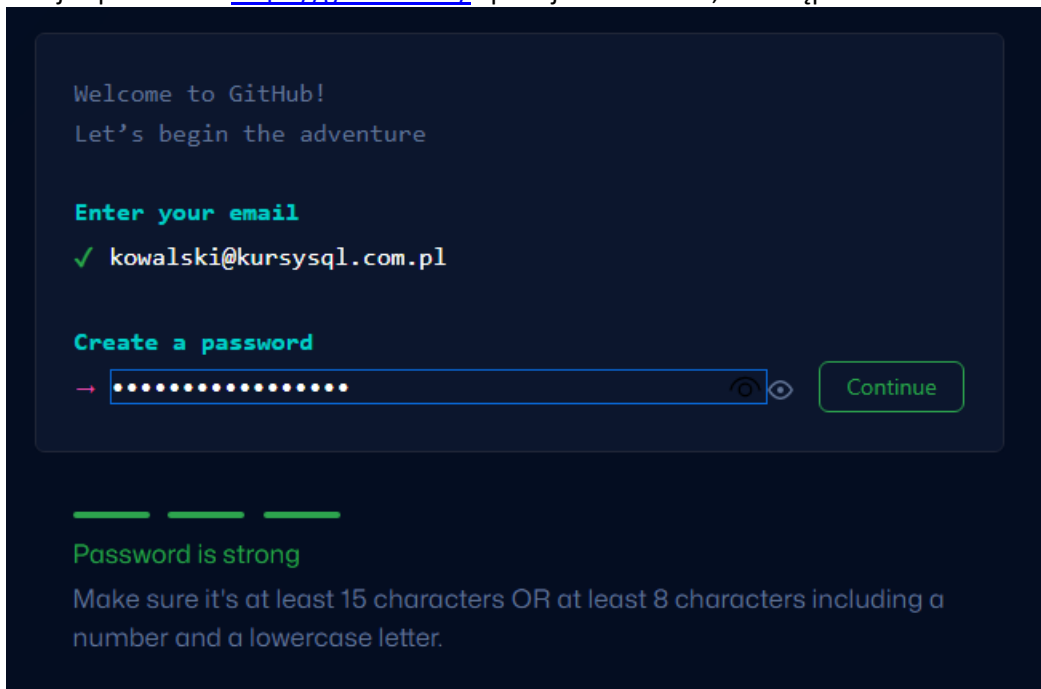
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   Tables/Aktor.sql
#   modified:   Tables/Kraj.sql
#
```

```
kowalski@amida MINGW64 /c/Repositories/Filmy (master)
$ git revert ae27b60
[master 1be697b] Revert "Cleanup tables definitions"
 2 files changed, 11 insertions(+), 9 deletions(-)
```

E. GitHub – wprowadzenie

1. Rejestracja konta

- a. Przejdź pod adres: <https://github.com/> i podaj adres e-mail, a następnie hasło



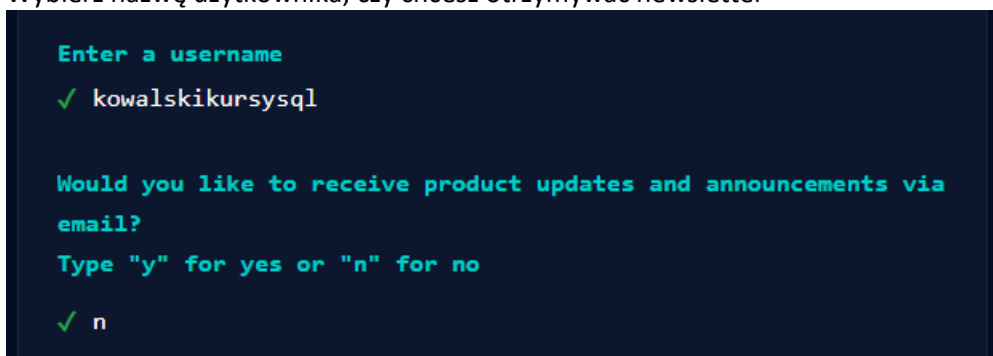
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ kowalski@kursysql.com.pl

Create a password
→ [password field] [eye icon] [Continue]

Password is strong
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

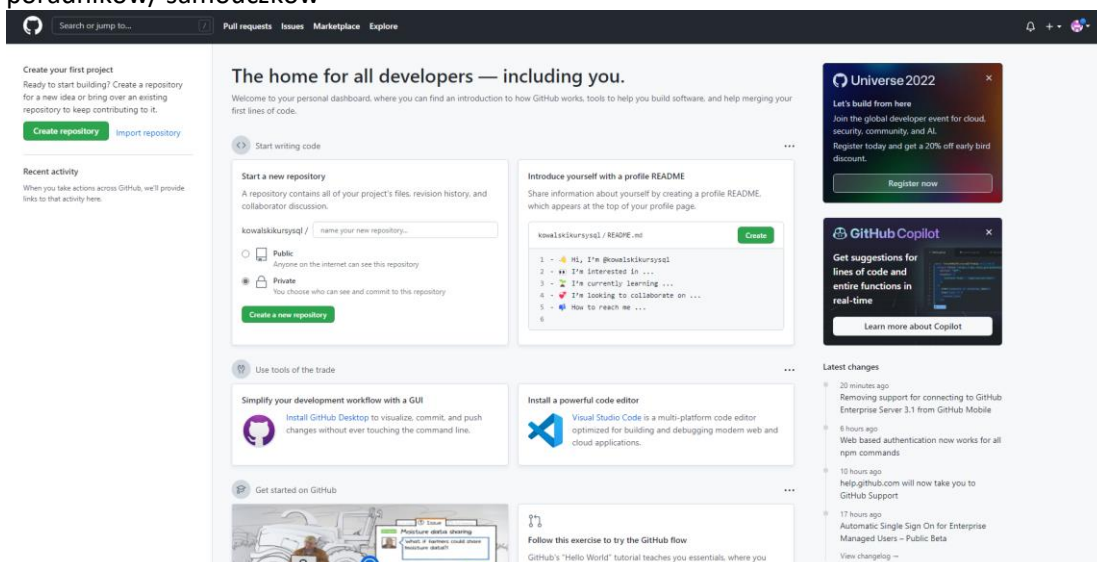
- b. Wybierz nazwę użytkownika, czy chcesz otrzymywać newsletter



Enter a username
✓ kowalskikursysql

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

- c. Będziesz musiał potwierdzić adres e-mail, a po utworzeniu konta zobaczysz stronę domową zawierającą możliwość łatwego utworzenia repozytorium, instalacji GitHub Desktop i wiele poradników/ samouczków



Search or jump to... Pull requests Issues Marketplace Explore

Create your first project
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.
Create repository Import repository

Recent activity
When you take actions across GitHub, we'll provide links to that activity here.

The home for all developers — including you.
Welcome to your personal dashboard, where you can find an introduction to how GitHub works, tools to help you build software, and help merging your first lines of code.

Start writing code

Start a new repository
A repository contains all of your project's files, revision history, and collaborator discussion.
kowalskikursysql / name your new repository...
Public Anyone on the internet can see this repository.
Private You choose who can see and commit to this repository.
Create a new repository

Introduce yourself with a profile README
Share information about yourself by creating a profile README, which appears at the top of your profile page.
kowalskikursysql / README.md
Create

Use tools of the trade

Simplify your development workflow with a GUI
Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

Install a powerful code editor
Visual Studio Code is a multi-platform code editor optimized for building and debugging modern web and cloud applications.

Get started on GitHub
Follow this exercise to try the GitHub flow
GitHub's "Hello World" tutorial teaches you essentials, where you create your own repository and learn GitHub's pull request workflow.

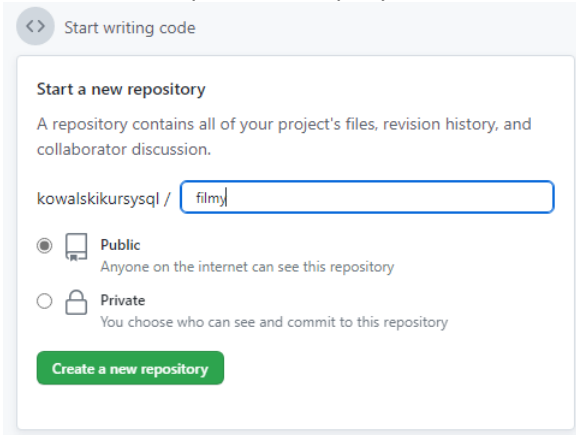
Universe2022
Let's build from here
Join the global developer event for cloud, security, community, and AI.
Register today and get a 20% off early bird discount.
Register now

GitHub Copilot
Get suggestions for lines of code and entire functions in real-time
Learn more about Copilot

Latest changes
20 minutes ago Removing support for connecting to GitHub Enterprise Server 3.1 from GitHub Mobile
6 hours ago Web based authentication now works for all npm commands
10 hours ago help.github.com will now take you to GitHub Support
17 hours ago Automatic Single Sign On for Enterprise Managed Users - Public Beta
View changelog

2. Tworzenie repozytorium i łączenie z lokalnym

a. Utwórz nowe – publiczne repozytorium o nazwie filmy



Start writing code

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

kowalskikursysql / filmy

☒ Public
Anyone on the internet can see this repository

☐ Private
You choose who can see and commit to this repository

Create a new repository

b. Zostanie wyświetlona strona zawierająca:

- url repozytorium
- gotowe polecenia, które spowodują utworzenie nowego, lokalnego repozytorium i połączenie go z tym utworzonym w ramach github
- polecenia pozwalające synchronizację istniejącego lokalnego repo z utworzonym filmy

c. Skorzystamy z tym ostatniej opcji, w tym celu w ramach git bash wykonaj kolejno:

`git status`

- aby upewnić się, że nie masz nic do zacomitowania

`git remote`

- aby połączyć lokalne repozytorium ze zdalnym

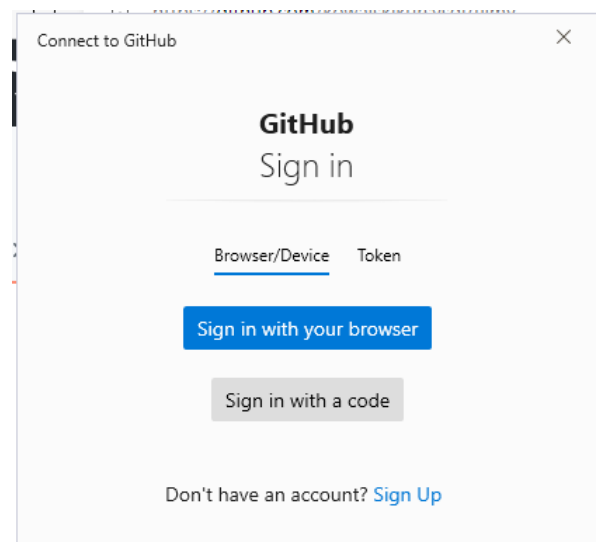
`git branch -M main`

- aby przełączyć się na główną gałąź (w naszym przypadku nie ma jeszcze innej)

`git push -u origin main`

- aby wysłać lokalne repozytorium do github

d. W ostatnim kroku pojawi się okno „Connect to GitHub”, wybierz **Sign in with your browser**, a następnie **Authorize GitCredentialManager**. Na końcu kreatora otrzymasz komunikat o poprawnym uwierzytelnieniu.



Connect to GitHub

GitHub

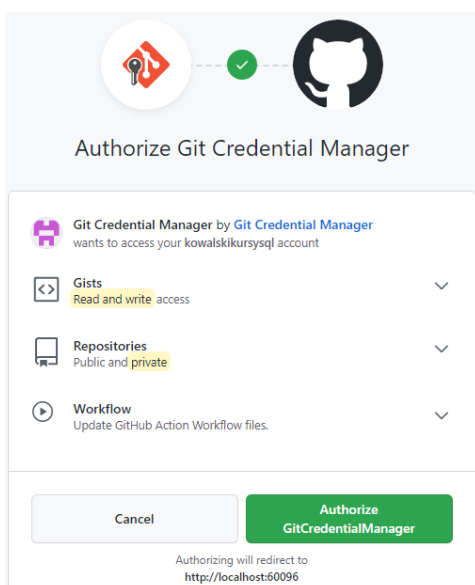
Sign in

Browser/Device Token

Sign in with your browser

Sign in with a code

Don't have an account? [Sign Up](#)



Authorize Git Credential Manager

Git Credential Manager by Git Credential Manager wants to access your kowalskikursysql account

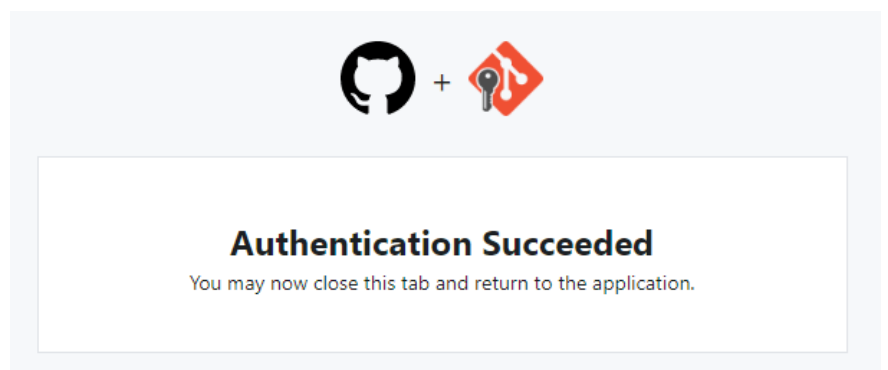
☒ Gists
Read and write access

☒ Repositories
Public and private

☒ Workflow
Update GitHub Action Workflow files.

Cancel Authorize GitCredentialManager

Authorizing will redirect to <http://localhost:60096>



Authentication Succeeded

You may now close this tab and return to the application.

- e. Twoje pliki zostaną przesłane do repozytorium w GitHub

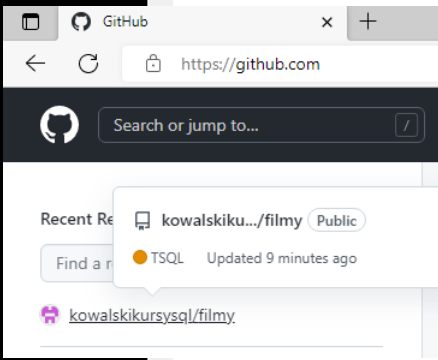
```
kowalski@amida MINGW64 /c/Repositories/Filmy (master)
$ git status
On branch master
nothing to commit, working tree clean

kowalski@amida MINGW64 /c/Repositories/Filmy (master)
$ git remote add origin https://github.com/kowalskikursysql/filmy.git

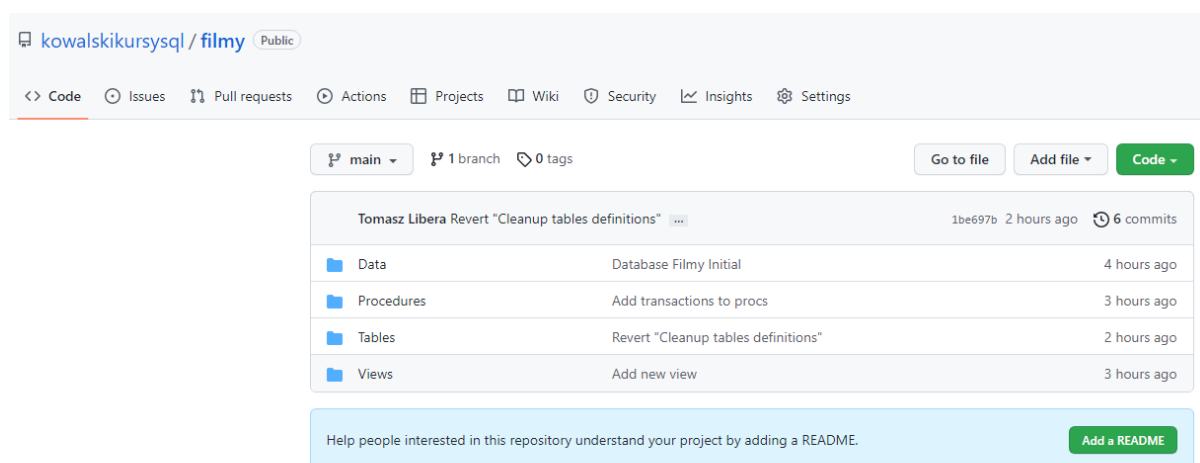
kowalski@amida MINGW64 /c/Repositories/Filmy (master)
$ git branch -M main

kowalski@amida MINGW64 /c/Repositories/Filmy (main)
$ git push -u origin main
Enumerating objects: 42, done.
Counting objects: 100% (42/42), done.
Delta compression using up to 8 threads
Compressing objects: 100% (40/40), done.
Writing objects: 100% (42/42), 10.54 KiB | 2.63 MiB/s, done.
Total 42 (delta 12), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (12/12), done.
To https://github.com/kowalskikursysql/filmy.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

kowalski@amida MINGW64 /c/Repositories/Filmy (main)
$
```

A screenshot of a web browser window showing the GitHub repository page for 'kowalskikursysql/filmy'. The browser's address bar shows 'https://github.com'. The page header includes the repository name and 'Public' status. Below the header, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the repository's file structure: Data, Procedures, Tables, and Views, each with a brief description and the time since the last commit. A 'Recent Revisions' section is partially visible on the left.

- f. Przejdź na stronę www.github.com – po lewej stronie zobaczysz odnośnik do Twojego pierwszego repozytorium (screen powyżej)



F. GitHub i praca grupowa

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

G. Narzędzia graficzne

(Git GUI, GitHubDesktop, Azure Data Studio)

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

H. Plik .gitignore

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

I. Plik README

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

J. Branches

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

K. Rozwiązywanie konfliktów

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>

L. Praca z projektami bazodanowymi i Git w Visual Studio i Azure Data Studio

Sprawdź pełną wersję szkolenia **SQL zaawansowany**

<https://www.kursysql.pl/szkolenie-sql-zaawansowany/>