

Samodzielność / proaktywność

BAZY DANYCH

• SQL — język deklaratywny (uczymy się myśleć)

- SQL Lite → SQLite Studio

- MySQL

- Access

preferowana
wersja → Portable

sqlite3

SQLAlchemy

bonus.

nasze_zwierzaki.db

[zapytanie]
query ~ prompty
kwerendy

```
CREATE TABLE osoby (  
    id INTEGER PRIMARY KEY,  
    imie TEXT,  
    nazw TEXT,  
    wiek INTEGER  
);
```

osoby

id	imie	nazw	wiek

```
DROP TABLE osoby;
```

```
INSERT INTO osoby (id, imie, nazw, wiek)  
VALUES (0, "Jarek", "Drapata", 48),  
       (1, "Arkadiusz", "Wojs", 56),  
       (2, "Steczowska", "Justyna", 52);
```

import faker

osoby

<u>id</u>	<u>imie</u>	<u>nazw</u>	<u>wiek</u>
0			

zapytanie_insert.txt

```
INSERT INTO osoby (id, imię, nazw, wiek)
VALUES (0, "Donata", "Rabczewska", 43),
(1, "Karol", "Strassburger", 81);
```

n_rekordow = 100

import faken

'z'.join(lista)

int(x)
str(x)

f"Wartość x = {x}"

SQLite Studio

DBeaver

DB Browser for SQL

CREATE TABLE IF NOT EXISTS osoby

Typowe błędy:

- nie ma połączenia z bazą,
- kilka działań w jednym „skrypcie”
- uruchamianie skryptu „na ślepo”
czyli bez kontroli całości
- wzależnianie się od innych,

```
lista = ['a', 'b']
```

```
zm_a, zm_b = *lista
```

```
lista.extend(druga_lista)  
?  
t
```

```
with open('zapytanie_insert.txt', 'w')  
as plik:
```

```
plik.write . . .
```

Generowanie sztucznych danych dla bazy

osoby

id	imie	nazw	wiek
0			



silnik bazodanowych

decyduje o
sposobie
wykonania
zapytania

W domu:

Wypróbować i
wybrać ulubione
narzędzie

SELECT * FROM osoby

SELECT imie, nazw FROM osoby

WHERE wiek > 60

SELECT * FROM osoby

WHERE nazw LIKE "S%"
"S%a"

WHERE imie LIKE "-----"
"N%a"

"%sk_"

• SQLite Studio

• DBeaver

• DB Browser for SQL

• ...

SELECT * FROM osoby

WHERE wiek != 18

<>

WHERE wiek BETWEEN 14 AND 19

WHERE wiek >= 14 AND wiek <= 19

WHERE wiek IN (10, 20, 30, 40)

NOT IN

Zad. NOT

lista zaproszeń z pominięciem nazwisk na „N”

Zad. OR

Osoby których imię lub nazw. na „A”

Zad. imię i nazw zakończone na inną literę

wersja 1

na tą samą

wersja 2

osoby

id	imie	nazw	wiek
----	------	------	------

"imie"

SELECT id, imie AS name, nazw AS surname,
wiek AS age

FROM osoby

ALTER TABLE osoby RENAME TO person
people

SELECT * FROM osoby
WHERE

filtr / kolumn

SELECT kolumny FROM tabela/e
WHERE filtr / warunki
rekordów

SELECT 2+3 AS WYNIK

SELECT ("zbychu" AS imie)

SELECT upper("zbychu") AS imie-

I sposób

generator1.py

dane.txt

(0, "Jan", "Himilbach", 62),
.....

II sposób

generator2.py

zapytanie.txt

INSERT INTO osoby (id, imie,)
VALUES (0, "Jan", "Himilbach", 62)
.....

UPDATE osoby SET nazw = upper(nazw)

UPDATE osoby SET nazw = lower(nazw), imie = lower(imie) || ". "
WHERE id = 8
|| "Δ" ||

z dużych, "zbych" AS imie_z_malych

SELECT id, imie || nazw AS godnosc
FROM osoby

osoby

id	imie	nazw	wiek
----	------	------	------

UPDATE osoby SET nazw = "+" || nazw
WHERE id IN

(
SELECT id

FROM osoby

WHERE wiek >= 18

)

DEKOMPOZYCYA ZADANIA
NA PROSTSZE PODZADANIA

ALTER TABLE osoby

ADD COLUMN plec TEXT

~~Ćw~~ Ćw

Uzupełnij kolumnę "plec" w-g reguły

ostatnia litera imienia to "a" - K

w przec. przyp. - M

DELETE FROM osoby WHERE imie = "Marek"

ALTER TABLE osoby DROP COLUMN plec

SELECT * FROM osoby

ORDER BY wiek

ORDER BY ~~nazw~~, imie DESC

imie, nazw

imie, wiek

imie DESC, wiek ASC

Mimesis

zwierzaki

domowy

id	imię	gat	wiek	żyje

def gen_zwierzaki(ile, plik)

nazwa
czy
zm.plikowe

plik = "moje_zwierz.txt"

?

plik = open(nazwa_pliku, 'w')

zwierzeta

id	imie	gatunek	wiek	zyje
...	...	...	...	...
8	Sunia	pies	3	'TAK'
...	...	...	...	...

Moje zajęcia to serial w odcinkach.
Jak opuszczam odcinek, to
oglądam w domu, co było.

SELECT * FROM zwierzeta

SELECT z.imie, z.gatunek, z.wiek
FROM zwierzeta AS z

SELECT COUNT(*) ^{AS ile} FROM zwierzeta

SELECT zyje, COUNT(*) ^{zyje} AS ile
FROM zwierzeta

funkcja agregująca

GROUP BY zyje

ORDER BY ile DESC
ASC

NULL

Nan

np. nan

None



SELECT gatunek, [↔]zysje, COUNT(*) as ile
FROM zwierzeta
GROUP BY gatunek, zysje
[↔]
ORDER BY gatunek DESC, zysje ASC

SELECT gatunek
FROM zwierzeta
GROUP BY gatunek

↓
SELECT DISTINCT gatunek
FROM zwierzeta

SELECT gatunek, COUNT(*) AS ile
FROM zwierzeta
GROUP BY gatunek

HAVING ile > 10

np.
ile > 10
⇒ HAVING COUNT(*) > 10,
w razie wątpliwości

Wyzwanie:

osiągnąć ten efekt bez HAVING

zwierzeta

id	imie	gatunek	wiek	zyje
...	...	...	...	...
8	Sunia	pies	3	'TAK'
...	...	...	...	...

```
SELECT imie, gatunek, wiek,  
CASE  
    WHEN wiek <= 1 THEN 'swiezak'  
    WHEN wiek <= 7 THEN 'dojrzone'  
    ELSE 'stare'  
END AS kat_wiek  
FROM zwierzeta  
WHERE zyje = 'TAK'
```