Rok, kierunek i grupa: 2023 Inżynieria Obliczeniowa, Grupa 3	**Temat:** Mini Projekt 3	**Data:** 03.12.2023
Przedmiot: Podstawy baz danych	**Imię i Nazwisko:** Filip Rak	**Ocena:**

Zadanie:

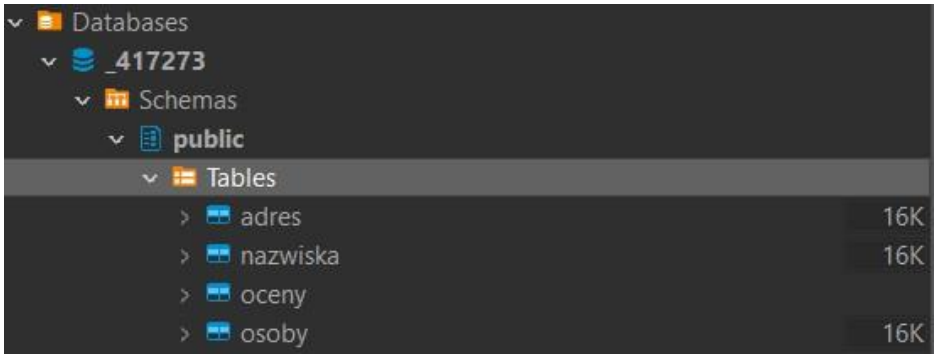
- Import danych zawartych w załączonych plikach .CSV
- Wykonanie zapytań wykorzystujących LEFT JOIN, RIGHT JOING INNER JOIN (3 tabele) i opisanie różnicy
- Utworzyć unię imion i nazwisk

Import danych zawartych w załączonych plikach:

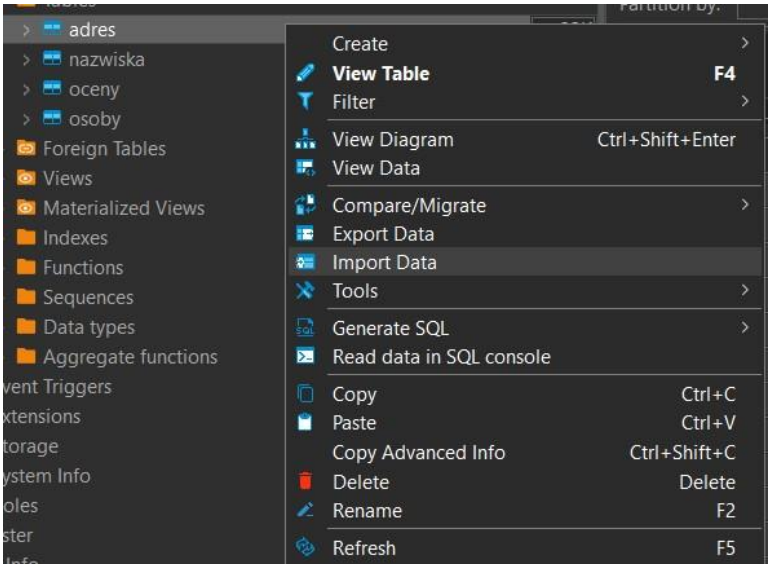
Przed rozpoczęciem importowania powinniśmy najpierw utworzyć tabele. Pierwszym krokiem zatem będzie otworzenie plików w edytorze tekstowym i przeanalizowanie ich pod kątem danych jakie przechowują. Następnie na podstawie tych danych tworzymy tabele

```
CREATE TABLE oceny (ID_osoby INT, Ocena INT);
CREATE TABLE adres (ID INT PRIMARY KEY, adres VARCHAR);
CREATE TABLE nazwiska (ID INT PRIMARY KEY, nazwisko VARCHAR);
CREATE TABLE osoby (ID INT PRIMARY KEY, imie VARCHAR, nazwisko INT, adres INT);
```

Jak widać na poniższym zrzucie ekranu, 4 tabele zostały utworzone



Następnym krokiem jest importowanie danych z plików .CSV do tabel



Jak można zobaczyć na poprzednich zrzutach ekranu, nadałem złą nazwę tabeli Adresy, poprawię ją poleceniem ALTER TABLE

```
ALTER TABLE adres RENAME TO adresy
```

Za pomocą polecenia SELECT sprawdzamy czy dane zostały poprawnie zaimportowane

```
SELECT * FROM adres
```

	id	adres
1	1	Adres1
2	2	Adres2
3	3	Adres3
4	4	Adres4
5	5	Adres5
6	6	Adres6
7	7	Adres7
8	8	Adres8
9	9	Adres9
10	10	Adres10
11	11	Adres11

```
SELECT * FROM nazwiska
```

id	nazwisko
1	Nazwisko1
2	Nazwisko2
3	Nazwisko3
4	Nazwisko4
5	Nazwisko5
6	Nazwisko6
7	Nazwisko7
8	Nazwisko8
9	Nazwisko9
10	Nazwisko10

```
SELECT * FROM oceny
```

id_osoby	ocena
1	3
2	2
3	4
4	1
5	4
6	5
7	2
8	4
9	2
10	6
11	4
12	3
1	4

```
SELECT * FROM osoby
```

id	imie	nazwisko	adres
1	Imie1	1	2
2	Imie2	1	2
3	Imie3	2	2
4	Imie4	3	1
5	Imie5	4	1
6	Imie6	4	1
7	Imie7	4	2
8	Imie8	1	2
9	Imie9	1	2
10	Imie10	2	2
11	Imie11	1	2
12	Imie12	2	2

Jak widać dane zostały zaimportowane poprawnie

Wykonanie zapytań wykorzystujących LEFT JOIN, RIGHT JOING INNER JOIN (3 tabele) i opisanie różnicy:

LEFT JOIN zwraca wszystkie wiersze z lewej tabeli, wypełniając brakujące dopasowania z prawej tabeli wartościami NULL.

```
SELECT osoby.id, imie, adresy.adres FROM osoby LEFT JOIN adresy ON osoby.adres = adresy.id
```

id	imie	adres
1	Imie1	Adres2
2	Imie2	Adres2
3	Imie3	Adres2
4	Imie4	Adres1
5	Imie5	Adres1
6	Imie6	Adres1
7	Imie7	Adres2
8	Imie8	Adres2
9	Imie9	Adres2
10	Imie10	Adres2
11	Imie11	Adres2
12	Imie12	Adres2

Jak widać w powyższym przykładzie, wszystkie dane są kompletne. Pokazuje nam to, że w wypadku w którym wszystko jest przypisane, efekt tego zapytania nie będzie się wizualnie różnić od INNER JOIN'a. Jednak jeżeli byśmy mieli osobę 13, która nie miała by przypisanego adresu to nadal była by ona wyświetlona, a w miejscu adresu byłby NULL.

RIGHT JOIN zwraca wszystkie wiersze z prawej tabeli, wypełniając brakujące dopasowania z lewej tabeli wartościami NULL. Innymi słowy jest to działanie odwrotne do LEFT JOIN

```
SELECT osoby.imie, adresy.adres FROM osoby RIGHT JOIN adresy ON osoby.adres = adresy.id
```

	imie	adres
10	Imie10	Adres2
11	Imie11	Adres2
12	Imie12	Adres2
13	[NULL]	Adres10
14	[NULL]	Adres8
15	[NULL]	Adres6
16	[NULL]	Adres7
17	[NULL]	Adres11
18	[NULL]	Adres5
19	[NULL]	Adres4
20	[NULL]	Adres3
21	[NULL]	Adres9

Jak widać na powyższym zrzucie ekranu, mamy znacznie więcej wyników niż w poprzednim zapytaniu. Dzieje się tak ponieważ, wiele z osób w naszej bazie danych mieszka razem. Tak właściwie to wszystkie osoby mieszkają pod Adresem1, lub Adresem2. Te osoby również zostały wszystkie wyświetlone, każda ze swoim przypisanym adresem. Jednak w naszej tabeli *Adresy* jest znacznie więcej rekordów. Ponieważ te adresy nie są do nikogo przypisane, są one wyświetlone z imieniem NULL. Naszym wnioskiem może być to, że każda osoba w naszej bazie ma przypisany adres, ale nie każdy adres ma przypisaną osobę. Warto dodać, że zmieniając kolejność argumentów w zapytaniu mielibyśmy taką samą sytuację co w przypadku LEFT JOIN.

INNER JOIN zwraca tylko te wiersze, które mają dopasowania w obu tabelach.

```
SELECT nazwiska.nazwisko, adresy.adres, imie
FROM osoby
INNER JOIN nazwiska ON nazwiska.id = osoby.nazwisko
INNER JOIN adresy ON adresy.id = osoby.adres
```

	nazwisko	adres	imie
1	Nazwisko1	Adres2	Imie1
2	Nazwisko1	Adres2	Imie2
3	Nazwisko2	Adres2	Imie3
4	Nazwisko3	Adres1	Imie4
5	Nazwisko4	Adres1	Imie5
6	Nazwisko4	Adres1	Imie6
7	Nazwisko4	Adres2	Imie7
8	Nazwisko1	Adres2	Imie8
9	Nazwisko1	Adres2	Imie9
10	Nazwisko2	Adres2	Imie10
11	Nazwisko1	Adres2	Imie11
12	Nazwisko2	Adres2	Imie12

W powyższym zapytaniu połączyliśmy trzy tabele za pomocą INNER JOIN: *Osoby, Nazwiska* i *Adresy*. Poprawnie używając INNER JOIN mamy gwarancje, że wyświetlone zostaną nam tylko te rekordy, które mają swoje odpowiedniki w innych tabelach. Jeśli wiersz z jednej tabeli nie ma odpowiadającego mu wiersza w drugiej tabeli, to nie zostanie on uwzględniony w wyniku. INNER JOIN jest najczęściej używanym typem połączenia.

Utworzyć unię imion i nazwisk:

```
SELECT imie AS "unia" FROM osoby
UNION
SELECT nazwisko AS "unia" FROM nazwiska
```

	unia
1	Imie5
2	Nazwisko2
3	Imie4
4	Nazwisko4
5	Imie11
6	Nazwisko6
7	Nazwisko10
8	Imie12
9	Nazwisko1
10	Nazwisko8
11	Imie2
12	Nazwisko7
13	Imie1

Polecenie UNION pozwala nam na wykonanie dwóch poleceń SELECT i wrzucenie wyników ich działania do pojedynczej kolumny. Dodatkowo, w tym zadaniu zdecydowałem się na zmianę nazwy wyświetlanej kolumny przy pomocy polecenia AS na „unia”, w celu lepszego opisania wyniku zapytania.