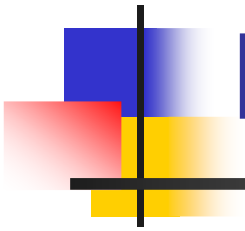
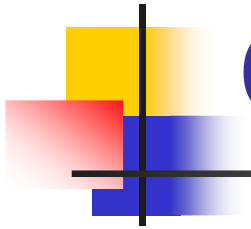


Bazy danych 2

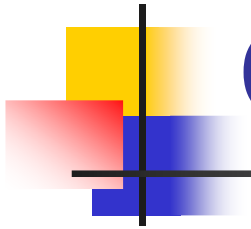
Wykład 4

Structured Query Language (SQL)



Cechy SQL

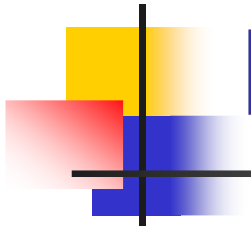
- W standardzie SQL wyróżnia się dwie części:
  - DDL *(Data Definition Language)* - język definiowania danych
  - DML (Data Manipulation Language) – język manipulowania danymi
- SQL jest językiem nieproceduralnym: użytkownik opisuje informacje, których potrzebuje, a nie wskazuje w jaki sposób należy ja odnaleźć
- Ma dość swobodny format – poszczególne fragmenty poleceń nie muszą być umieszczone w określonych miejscach
- SQL nie jest językiem zupełnym obliczeniowo
- SQL może być osadzony w języku proceduralnym



Cechy SQL

Będziemy UŻYWAĆ SKŁADNI Backusa-Naura

- Wielkie litery będziemy używać w poleceniach
- Małymi litrami będziemy zapisywać sowa definiowane przez użytkownika
- Pionowej kreski będziemy używać , by zaznaczyć możliwość wyboru jednej z przedstawionych opcji
- W nawiasach klamrowych umieszczamy elementy wymagane
- W nawiasach kwadratowych umieszczamy elementy opcjonalne
- Nawiasów okrągłych (…) używamy, aby zaznaczyć możliwość powtórzenia elementu zero lub dowolna liczbę razy
- Literały – to stałe wykorzystywane w poleceniach SQL, nieliczbowe wartości danych zapisujemy w apostrofach, a wartości liczbowe bez



Klauzula SELECT

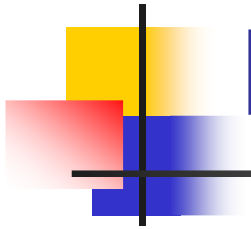
```
SELECT      [DISTINCT|ALL]{*|[wyrażenie kolumnowe[AS nowa_nazwa]] [,…]}
FROM        NazwaTabeli [alias][,…]
[WHERE                  warunek_selekcji wierszy]
[GROUP BY   lista_kolumn][HAVING warunek_selekcji_grup]
[ORDER BY   lista_kolumn];
```

Klauzula SELECT

Kolejność przetwarzania jest następująca

- FROM — określa tabelę (lub tabele), z których będziemy korzystać
- WHERE — pozwala wybrać wiersze spełniające zadany warunek selekcji wierszy
- GROUP BY — tworzy grupy wierszy o tej samej wartości wskazanej kolumny
- HAVING — pozwala wybrać grupy ze względu na podany warunek selekcji
- SELECT — wskazuje, które kolumny powinny pojawić się w wyniku
- ORDER BY — określa uporządkowanie wyniku

☝ Porządku elementów zapytania SELECT nie można zmieniać



Klauzula SELECT

Przykład 1. Wyszukiwanie wszystkich kolumn i wierszy

Podaj wszystkie dane wszystkich pracowników

```
SELECT  *
FROM Personel;
```

Przykład 2. Wyszukiwanie wybranych kolumn i wszystkich wierszy

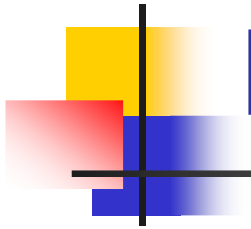
Podaj listę płac wszystkich pracowników, lista powinna zawierać jedynie numer pracownika, jego imię i nazwisko oraz pensję.

```
SELECT pracownikNr, imię, nazwisko, pensja
FROM Personel;
```

Przykład 3. Wykorzystanie DISTINCT

Podaj numery wszystkich nieruchomości, które zostały odwiedzone przez klientów

```
SELECT DISTINCT nieruchomośćNr
FROM Wizyta;
```



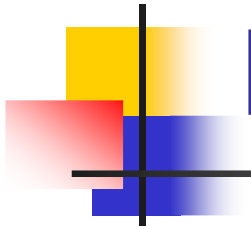
Pola wyliczane

Przykład 4. Pola wyliczane

Podaj listę miesięcznych płac wszystkich pracowników; lista powinna zawierać numer pracownika , jego imię i nazwisko oraz płacę (w tabeli pensje są pensjami rocznymi)

```
SELECT  PracownikNr, imię, nazwisko, pensja/12 AS pensjaMiesięczna
FROM Personel;
```

- Polecenie SQL może zawierać dodawanie, odejmowanie, mnożenie i dzielenie oraz nawiasy pozwalające budować bardziej skomplikowane wyrażenia
- W kolumnie wyliczanej może wystąpić więcej niż jedna kolumna



Klauzula WHERE

- Klauzula WHERE zawiera warunek selekcji, 5 podstawowych warunków to:
  - *Porównanie* – polega na porównaniu wartości jednego wyrażenia z wartością drugiego wyrażenia
  - *Sprawdzenie zakresu* – polega na sprawdzeniu, czy zadana wartość należy do wskazanego przedziału wartości
  - *Przynależność do zbioru* – polega na sprawdzeniu, czy zadana wartość jest równa jednemu spośród elementów zbioru
  - *Dopasowanie do wzorca* – polega na sprawdzeniu czy słowo pasuje do podanego wzorca
  - *Wartośc pusta* – polega na sprawdzeniu, czy w kolumnie jest wartość pusta

Klauzula WHERE

Przykład 5. Warunek selekcji: porównanie

Podaj wszystkich pracowników, których pensja jest wyższa niż 10000 funtów

```
SELECT  pracownikNr, imię, nazwisko, stanowisko, pensja
FROM Personel
WHERE pensja>10000;
```

Przykład 6. Złożony warunek selekcji: porównanie

Podaj adresy wszystkich biur znajdujących się w Londynie lub Glasgow

```
SELECT *
FROM Biuro
WHERE miasto='Londyn' OR miasto='Glasgow';
```

Klauzula WHERE

Przykład 6. Warunek selekcji: wartości z zakresu (BETWEEN i NOT BETWEEN)

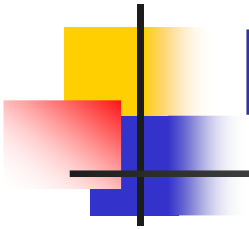
Podaj wszystkich pracowników mających pensję pomiędzy 20000 a 30000 funtówww

```
SELECT  pracownikNr, imię, nazwisko, stanowisko, pensja
FROM Personel
WHERE pensja BETWEEN 20000 AND 30000;
```

Przykład 7. Warunek selekcji : przynależność do zbioru (IN lub NOT IN)

Podaj wszystkich kierowników i dyrektorów

```
SELECT  pracownikNr, imię, nazwisko, stanowisko
FROM Personel
WHERE stanowisko IN (‘kierownik’, ‘dyrektor’);
```



Klauzula WHERE

- W SQL występują dwa szczególne symbole zastępcze:
  - % - znak procentu zastępuje dowolny ciąg znaków
  - _ - znak podkreślenia zastępuje dowolny (jeden) znak

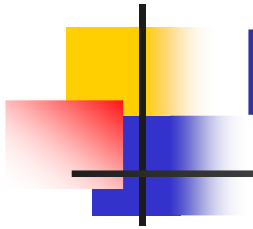
Przykład 8. Warunek selekcji: dopasowanie do wzorca (LIKE lub NOT LIKE)

Podaj wszystkich właścicieli, w których adresie występuje słowo ‘Glasgow’’

```
SELECT  klientNr, imię, nazwisko, adres, telNr
FROM WłaścicielPryatny
WHERE adres LIKE ‘%Glasgow%’;
```

- Jeśli słowo powinno zawierać znak specjalny musimy użyć klauzuli ESCAPE, np. by znaleźć ciąg ’15%’ użyjemy warunku

  LIKE ’15#%’ ESCAPE ‘#’



Klauzula WHERE

Przykład 9. Warunek selekcji: wartości puste (IS NULL lub IS NOT NULL)

Podaj szczegółowe informacje o wszystkich wizytach w nieruchomości PG4, po których nie zgłoszono uwag

```
SELECT  klientNr, dataWizyty
FROM Wizyta
WHERE nieruchomośćNr='PG4' AND uwagi IS NULL;
```

Klauzula ORDER BY

- Do uporządkowania wierszy będących wynikiem zapytania służy klauzula ORDER BY zawierająca listę oddzielonych przecinkami identyfikatorów kolumn, wg których należy posortować wynik

Przykład 10. Porządkowanie według jednej kolumny

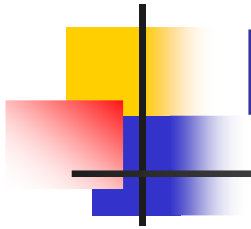
Wygeneruj listę pensji wszystkich pracowników uporządkowaną malejąco według pensji

```
SELECT  pracownikNr, imię, nazwisko, pensja
FROM Personel
ORDER BY pensja DESC
```

Przykład 11. Porządkowanie według wielu kolumn**.**

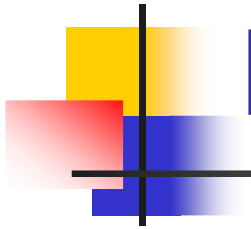
Wygeneruj listę wybranych informacji dotyczących nieruchomości uporządkowana według rodzajów nieruchomości

```
SELECT  nieruchomośćNr, typ, pokoje, czynszimię, nazwisko, pensja
FROM Nieruchomość
ORDER BY typ, czynsz DESC;
```



Funkcje agregujące

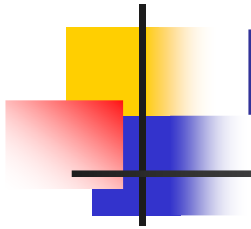
- W standardzie SQL pięć funkcji agregujących
  - COUNT – zwraca liczbę wartości występujących w określonej kolumnie
  - SUM – zwraca sumę wartości występujących w określonej kolumnie
  - AVG – zwraca średnią wartości występujących w określonej kolumnie
  - MIN – zwraca najmniejsza wartość występującą w określonej kolumnie
  - MAX – zwraca największą wartość występującą w określonej kolumnie
- Funkcje COUNT, MIN i MAX można stosować zarówno do wartości liczbowych jak i nieliczbowych
- Wszystkie funkcje oprócz COUNT(*) pomijają wartości puste
- Jeżeli chcemy wyeliminować powtórzenia, używamy słowa kluczowego DISTINCT przed nazwą kolumny w argumencie funkcji, może być użyte tylko jeden raz w zapytaniu



Funkcje agregujące

- Funkcje agregujące mogą być stosowne jedynie na liście SELECT lub klauzuli HAVING
- Jeżeli lista SELECT zawiera funkcję agregującą i w zapytaniu nie jest zastosowana klauzula GROUP BY służąca do grupowania danych, to wówczas żaden z elementów listy SELECT nie może odwoływać się do kolumny, o ile ta kolumna nie jest argumentem funkcji agregującej, np. błędne jest poniższe zapytanie

```
SELECT pracownikNr, COUNT(pensja)
FROM Personel
```



Funkcje agregujące

Przykład 12. Zastosowanie COUNT(*)

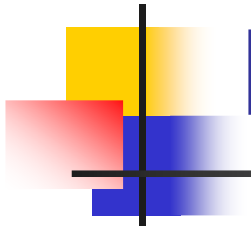
W ilu nieruchomościach miesięczny czynsz jest wyższy niż 350 funtów.

```
SELECT  COUNT(*) AS liczba
FROM Nieruchomość
WHERE czynsz>350;
```

Przykład 13. Zastosowanie COUNT DISTINCT

Ile nieruchomości odwiedzono w maju 2001 roku?

```
SELECT  COUNT(DISTINCT nieruchomościNr) AS liczba
FROM Wizyta
WHERE data Wizyty BETWEN ‘1.05.2001” AND ’31.05.2001’;
```



Funkcje agregujące

Przykład 14. Zastosowanie COUNT i SUM

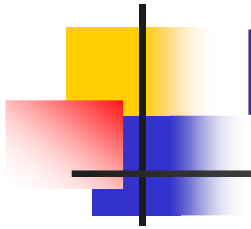
Oblicz, ilu jest dyrektorów i jaka jest ich sumaryczna pensja.

```
SELECT  COUNT(pracownikNr) AS liczba, SUM(pensja) as suma
FROM Personel
WHERE stanowisko='dyrektor';
```

Przykład 15. Zastosowanie MIN, MAX i AVG

Oblicz najmniejszą, największa i średnią pensję pracownika

```
SELECT  MIN(pensja) S minimum, MAX(pensja) AS maksimum, AVG(pensja)
AS średnia
FROM Personel;
```



Klauzula GROUP BY

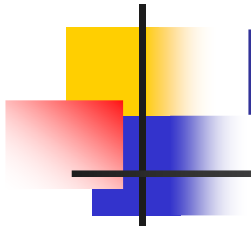
- Zapytanie z klauzulą GROUP BY nazywamy zapytaniem grupującym, ponieważ w trakcie jego obliczania dane z tabeli SELECT są dzielone na grupy i dla każdej grupy jest generowany jeden wiersz podsumowania.
- Kolumny wymienione w klauzuli GROUP BY nazywamy kolumnami grupującymi.
- Gdy w zapytaniu występuje GROUP BY, dla każdego elementu z listy SELECT musi istnieć możliwość wyznaczenia jednoznacznie wartości w ramach grupy
- Klauzula SELECT może zawierać jedynie
  - Nazwy kolumn grupowania
  - Funkcje agregujące
  - Stałe
  - Wyrażenia zawierające kombinacje powyższych elementów
- Wszystkie nazwy kolumn na liście SELECT muszą występować w klauzuli GROUP BY, chyba że nazwa kolumny jest używana jako argument funkcji agregującej

Klauzula GROUP BY

Przykład 16. Zastosowanie GROUP BY

Oblicz, dla każdegobiura liczbę zatrudnionych w nim pracowników oraz ich sumaryczna pensję.ilu jest dyrektorów i jaka jest ich sumaryczna pensja.

```
SELECT  biuroNr, COUNT(pracownikNr) AS liczba, SUM(pensja) AS suma
FROM Personel
GROUP BY biuroNr
ORDER BY biuroNr;
```



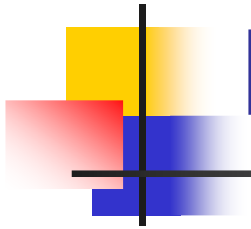
Klauzula HAVING

- Klauzulę HAVING stosuje się razem w połączeniu z klauzulą GROUP BY
- Klauzula HAVING służy do wyboru grup, które ostatecznie trafia do tabeli wynikowej
- Nazwy kolumn występujące w klauzuli HAVING pojawiały się także na liście GROUP BY lub były argumentami funkcji agregującej

Przykład 17. Zastosowanie HAVING

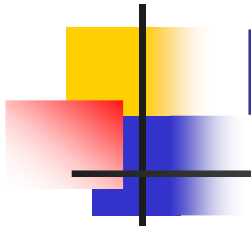
Dla każdego biura zatrudniającego więcej niż jednego pracownika, podajliczbę pracowników biura oraz sumę ich zarobków

```
SELECT  biuroNr, COUNT(pracownikNr) AS liczba, SUM(pensja) AS suma
FROM Personel
GROUP BY biuroNr
HAVING COUNT(pracownikNr)>1
ORDER BY biuroNr;
```



Podzapytania

- Zapytania zagnieżdżone (wewnętrzne) mogą występować w klauzulach WHERE i HAVING zapytania zewnętrznego, a także w klauzulach INSERT, UPDATE i DELETE
- Istnieją trzy rodzaje podzapytań:
  - *Podzapytania skalarne* – zwracają jedną kolumnę i jeden wiersz
  - *Podzapytania krotkowe* – zwracają kilka kolumn i tak jak poprzednio tylko jeden wiersz
  - *Podzapytanie tabelowe* – zwracają jedną lub więcej kolumn i wiele wierszy



Podzapytania

Przykład 18. Zastosowanie podzapytania z równością

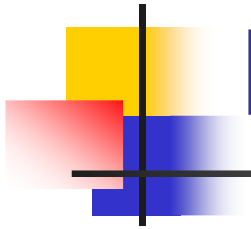
Podaj wszystkich pracowników zatrudnionych w biurze przy '163 MainStr.'

```
SELECT  pracownikNr, imię, nazwisko, stanowisko
FROM Personel
WHERE biuroNr=(SELECT biuroN
                    FROM Biuro
                     WHERE ulica='163 Main Str.');
```

Przykład 19. Stosowanie podzapytań z funkcją agregującą

Podaj wszystkich pracowników, których pensja jest wyższa od średniej; pokaż różnicę między poszczególnymi pensjami z średnią.

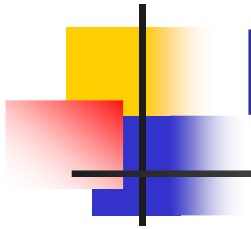
```
SELECT  pracownikNr, imię, nazwisko, stanowisko, pensja – różnica
FROM Personel
WHERE pensja>(SELECT AVG(pensja)   FROM Personel AS różnica) ;
```



Podzapytania

Do podzapytań stosuje się poniższe zasady:

- W podzapytaniach nie wolno używać klauzuli ORDER BY
- Lista SELECT podzapytania musi składać się z pojedynczej nazwy kolumny lub wyrażenia, z wyjątkiem podzapytań wykorzystywanych z operatorem EXISTS
- Domyślnie nazwy kolumn w podzapytaniu odnoszą się do nazwy tabeli z klauzuli FROM podzapytania. Do kolumn tabeli z klauzulą FROM zapytania zewnętrznego można odwołać się poprzedzając nazwę kolumny nazwą tabeli.
- Jeżeli podzapytanie jest jednym z dwóch argumentów, których dotyczy porównanie, to musi występować po prawej stronie porównania.



Podzapytania

Przykład 20. Podzapytania zagnieżdżone:

Podaj wszystkie nieruchomości nadzorowane przez pracowników zatrudnionych w biurze ‘163 Main Str’

```
SELECT nieruchomośćNr, ulica, miasto, kodPocztowy, typ, pokoje, czynsz
FROM Nieruchomość
WHERE pracownikNr IN (SELECT  pracownikNr
                      FROM Personel
                      WHERE biuroNr=(SELECT biuroN
                                     FROM Biuro
                                     WHERE ulica=‘163 Main Str.’));
```

Klauzule ANY, SOME i ALL

- Słowa ANY, SOME i ALL w przypadku gdy wynik daje wiele wierszy i stosujemy operatory porównania

Przykład 20. Zastosowanie ANY/SOME:

Znajdź wszystkich pracowników, którzy mają pensję wyższa niż przynajmniej jeden pracownik biura o numerze B003

```
SELECT pracownikNr, imię, nazwisko, pensja
FROM Personel
WHERE pensja>SOME (SELECT  pensja
                   FROM Personel
                   WHERE biuroNr='B003');
```

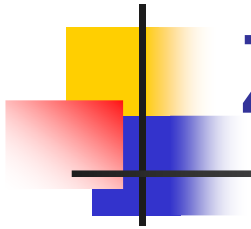
Zapytania dotyczące wielu tabel

- Aby dokonać złączenia w klauzuli FROM należy wymienić tabele oddzielając je przecinkami, a w klauzuli WHERE określić kolumny wg których jest dokonywane złączenie
- Dla każdej tabeli można zdefiniować alias – można go używać wszędzie zamiast nazwy

Przykład 22. Proste złączenie:

Podaj nazwy wszystkich klientów, którzy odwiedzili wszystkie nieruchomości . Wraz z danymi klienta podaj zgłoszone przez niego uwagi.

```
SELECT k.klientNr, imię, nazwisko, nieruchomośćNr, uwagi
FROM Klient k, Wizyta w
WHERE k.klientNr=w.klientNr
```



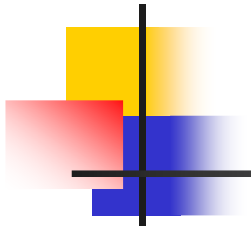
Zapytania dotyczące wielu tabel

- W standardzie SQL opisano alternatywne sposoby zapisu powyższego złaczenia

```
FROM Klient k JOIN Wizyta w ON k.klientNr=w.klientNr

FROM Klient JOIN Wizyta USING klientNr

FROM Klient NATURAL JOIN Wizyta
```



Złączenia

Procedura generowania wyniku zapytania SELECT ze złączeniem jest następująca:

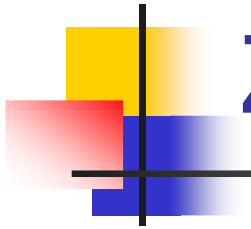
(1) Utwórz iloczyn kartezjański tabel wymienionych po klauzuli FROM

(2) Jeżeli istnieje klauzula WHERE, to zastosuj warunek selekcji do każdego z wierszy iloczynu, pozostawiając tylko te wiersze, które spełniają warunek

(3) Dla każdego z pozostałych wierszy ustal wartość każdego elementu z listy SELECT i wygeneruj jeden wiersz

(4) Jeżeli w zapytaniu użyto SELECT DISTINCT, usuń powtarzające sie wiersze z tabeli wynikowej

(5) Jeżeli występuje klauzula ORDER BY, to uporządkuj tabelę wynikową według ustalonego kryterium



Złączenia zewnętrzne

- Złączenie zewnętrzne zachowuje wiersze niespełniające warunku złączenia

- Istnieją trzy rodzaje złączenia zewnętrznego:
  - Lewostronne – LEFT JOIN
  - Prawostronne – RIGHT JOIN
  - Pełne – FULL JOIN

Złączenia zewnętrzne

Rozważmy przykład:

Biuro1

biuroNr	bmiasto
B003	Glasgow
B004	Bristol
B002	Londyn

Nieruchomość1

nieruchomośćNr	nmiasto
PA14	Aberdeen
PL94	Londyn
PG4	Glasgow

Złączenie lewostronne:

```
SELECT b.*, d.*
FROM biuro1 b LEFT JOIN Nieruchomość1 d
ON b.bmiasto=d.nmiasto;
```

biuroNr	bmiasto	nieruchomośćNr	nmiasto
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	Londyn	PL94	Londyn

Złączenie prawostronne:

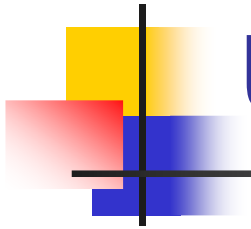
```
SELECT b.*, d.*
FROM biuro1 b RIGHT JOIN Nieruchomość1 d
ON b.bmiasto=d.nmiasto;
```

biuroNr	bmiasto	nieruchomośćNr	nmiasto
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	Londyn	PL94	Londyn

Złączenie pełne:

```
SELECT b.*, d.*
FROM biuro1 b FULL JOIN Nieruchomość1 d
ON b.bmiasto=d.nmiasto;
```

biuroNr	bmiasto	nieruchomośćNr	nmiasto
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	Londyn	PL94	Londyn



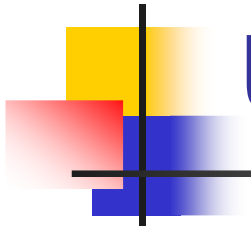
UNION, INTERSECT, EXCEPT

- *W języku SQL mamy możliwość wykonywania operacji na zborach: sumy, iloczynu, różnicy*
- *Operacje te nazywane są w standardzie SQL odpowienio: UNION, INTERSECT i EXCEPT*
- *Sposób zapisu jest następujący*

  operacja [ALL] [CORRESPONDING [BY [kolumna1[,...]]]]

  gdzie

  - *opcja CORRESPONDING [BY [kolumna1[,...]]] informuje, że operacja jest wykonywana na wskazanych kolumnach*
  - *opcja CORRESPONDING informuje, że operacja jest wykonywana w oparciu o wspólne kolumny*
  - *opcja ALL - wynik może zawierać powtarzające się wiersze*



UNION, INTERSECT, EXCEPT

Przykład 25. Zastosowanie UNION:

Podaj listę wszystkich miast w których znajduje się biuro lub nieruchomość

```
(SELECT miasto
FROM biuro
WHERE miasto IS NOT NULL)
UNION
(SELECT miasto
FROM Nieruchomość
WHERE miasto IS NOT NULL)
```

lub

```
(SELECT *
FROM biuro
WHERE miasto IS NOT NULL)
UNION CORRESPONDING BY miasto
(SELECT *
FROM Nieruchomość
WHERE miasto IS NOT NULL)
```

Dziękuję za uwagę