Bazy danych
Laboratorium 2
Info

Funkcje operujące na ciągach znaków:

- ***Concat****(str1, str2)* - zwraca ciąg znaków będący złączeniem ciągu *str1* z ciągiem *str2*,
- ***Initcap****(str)* - zwraca sformatowany ciąg wejściowy, tak, że pierwsza litera jest duża a pozostałe są małe,
- ***Instr****(str1, str2 [,pos [, n]])* - zwraca pozycje ciągu znaków *str2*, w *str1*, szukanie rozpoczyna się od *pos*, a kończy na *n*-tym wystąpieniu szukanego ciągu. Jeśli *pos*< 0 przeszukiwanie przebiega od końca. Jeśli ciąg *str2* nie zostanie znaleziony to funkcja zwraca wartość 0,
- ***Length****(str)* - zwraca długość *str*,
- ***Lower****(str)* - zwraca łańcuch *str* przekształcony na małe litery,
- ***Lpad****(str, len [, pad_str])* - uzupełnia z lewej strony ciąg *str* ciągiem *pad_str*, aż do uzyskania przez ciąg wynikowy długości *len*; domyślnie *pad_str* jest spacją,
- ***Ltrim****(str [, set_str])* - poczynając z lewej strony odcina kolejne znaki ciągu *str*, aż do napotkania pierwszego znaku nie należącego do zbioru *set_str* (domyślnie spacja),
- ***Replace****(str1, str2 [, str3])* - zwraca ciąg *str1* ze wszystkimi wystąpieniami ciągu *str2* zastąpionymi przez ciąg *str3;* brak *str3* oznacza usuniecie *str2*,
- ***Rpad****(str, len [, pad_str])* - uzupełnia z prawej strony ciąg *str* ciągiem *pad_str*, aż do uzyskania przez ciąg wynikowy długości *len*; domyślnie *pad_str* jest spacją,
- ***Rtrim****(str [, set_str])* - poczynając z prawej strony odcina kolejne znaki ciągu *str*, aż do napotkania pierwszego znaku nie należącego do zbioru *set_str* (domyślnie spacja),
- ***Substr****(str, pos [,len])* - zwraca podciąg ciągu *str* o długości *len* zaczynając od *pos*; jeśli *pos*< *0* przeszukiwanie przebiega od końca; domyślnie przebiega do końca,
- ***Trim****([[LEADING|TRAILING|BOTH] str1 FROM] str2)* - usuwa pierwsze wystąpienia ciągu *str1* w ciągu *str2*, zaczynając od początku - *LEADING*, końca - *TRAILING*, bądź z obu stron - *BOTH*; domyślnie obcinane są spacje z obu stron,
- ***Upper****(str)* - przekształca wszystkie litery w *str* na duże.

Przykłady

	PERSONALIA
1	Baranowski Gerard
2	Bednarczyk Cezary
3	Romanowski Bronisław

Rys. 1 Fragment wyniku zwróconego przez zapytanie:
*select Concat(nazwisko, **Concat**(' ', imiona)) Personalia from studenci;*

	NAZWISKO
1	Krol
2	Michalski
3	Kukulski

Rys. 2 Fragment wyniku zwróconego przez zapytania:
*SELECT **Initcap**(nazwisko) Nazwisko FROM pracownicy;*
*SELECT **Concat(**Upper(Substr(nazwisko,1,1)), Lower(Substr(nazwisko,2,Length(nazwisko)-1))**)** Nazwisko FROM pracownicy;*

	IMIONA	A	B	C	D
1	Adam	dam	X am	123456Adam	M
2	Adolf	dolf	X olf	12345Adolf	OLF
3	Adrian	drian	X rian	1234Adrian	IAN
4	Agata	gat	Agata	12345Agata	TA
5	Agnieszka	gnieszk	Agnieszka	1Agnieszka	IESZKA
6	Albert	lbert	Albert	1234Albert	LBERT
7	Albin	lbin	Albin	12345Albin	LBIN

Rys. 3 Fragment wyniku zwróconego przez zapytanie:
*SELECT distinct(imiona), **Trim**(BOTH 'a' FROM lower(imiona)) A, **Replace**(imiona,'Ad','X_') B,*
*__Lpad__(imiona,10,'123456789') C, **Ltrim**(upper(imiona), 'NDARG') D FROM studenci order by 1;*

Funkcje matematyczne:

- ***Abs(n)*** - zwraca wartość bezwzględna z *n*,
- ***Ceil(n)*** - zwraca najmniejsza liczbę całkowitą większą lub równa *n*,
- ***Cos(n)*** - zwraca cosinus kata *n* (wyrażonego w radianach),
- ***Floor(n)*** - zwraca największa liczbę całkowita mniejsza lub równa *n*,
- ***Ln(n)*** - zwraca logarytm naturalny z *n*,
- ***Log(n, m)*** - zwraca logarytm z liczby *m* przy podstawie *n*,
- ***Mod(m, n)*** - zwraca resztę z dzielenia liczby *m* przez *n*,
- ***Power(m, n)*** - zwraca wartość liczby *m* podniesionej do potęgi *n*,
- ***Round(m [, n])*** - zwraca liczbę *m* zaokrągloną do *n* miejsc po przecinku; dla n < 0, określa liczbę miejsc przed przecinkiem; domyślnie *n* = 0,
- ***Sign(n)*** - zwraca −1 dla *n* < 0, 0 dla *n* = 0 oraz 1 dla *n* > 0,
- ***Sin(n)*** - zwraca sinus kata *n* (wyrażonego w radianach),
- ***Sqrt(n)*** - zwraca pierwiastek kwadratowy z liczby *n*,
- ***Tan(n)*** - zwraca tangens kata *n* (wyrażonego w radianach),
- ***Trunc(m [,n])*** - zwraca liczbę *m* po obcięciu do *n* miejsc po przecinku; dla *n* < 0, określa liczbę miejsc przed przecinkiem; domyślnie *n* = 0.

Wybrane funkcje wierszowe:

- ***Decode(wyr, s1, r1, s2, r2, ... [,wyr_domyślne])*** - porównuje wyrażenie *wyr* do kolejnych *s1*, *s2*, ... Jeżeli *wyr* jest równe któremuś z tych wyrażeń, w wyniku zwracana jest odpowiadająca jemu wartość *ri*. Jeżeli żadna z wartości *s1*, *s2*, ... nie jest równa wejściowemu wyrażeniu funkcja zwraca wartość domyślną (*wyr_domyślne*) - jeśli jest ona pominięta wówczas funkcja zwraca wartość NULL.
- ***Nvl(wyr1, wyr2)*** - jeżeli *wyr1* ma wartość NULL, to funkcja zwraca *wyr2*, w przeciwnym przypadku funkcja zwraca *wyr1*.
- ***Uid*** - zwraca unikalny numer użytkownika,
- ***User*** - zwraca nazwę użytkownika.

	NAZWISKO	ID_DZIALU	INFO
1	KROL	10	KROL pracuje w D10
2	MICHALSKI	40	MICHALSKI pracuje w innym dziale
3	KUKULSKI	30	KUKULSKI pracuje w D30
4	WIERZBICKI	40	WIERZBICKI pracuje w innym dziale

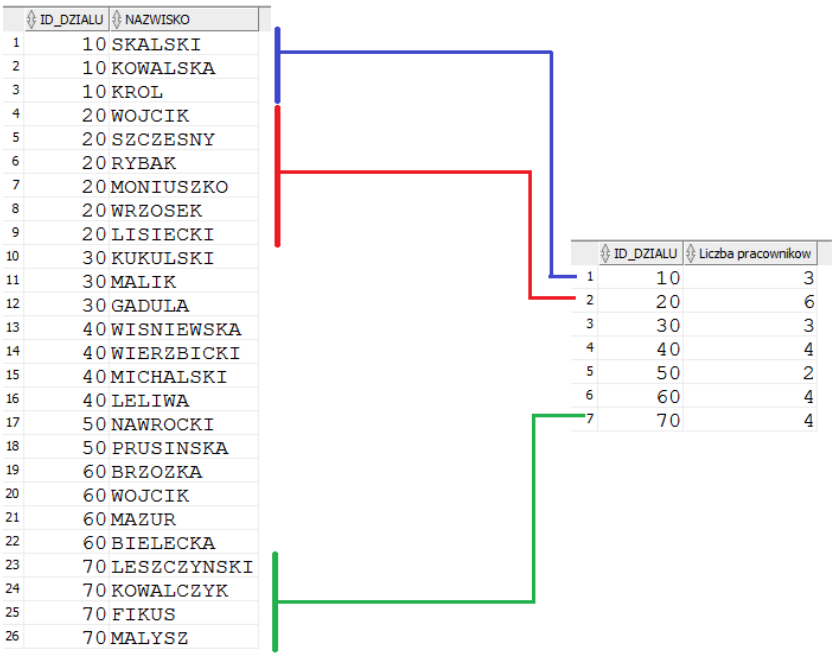
Rys. 4 Fragment wyniku zwróconego przez zapytanie:
SELECT nazwisko, id_dzialu, **Decode(**id_dzialu, **10**, nazwisko||' pracuje w D10', **30**, nazwisko||' pracuje w D30', nazwisko||' pracuje w innym dziale'**)** INFO FROM pracownicy;

Funkcje agregujące:

- ***Avg([DISTINCT] wyr)*** - oblicza średnią ze zbioru (opcjonalnie różnych) wartości wyrażenia (pomijając wystąpienia NULL),
- ***Count(*|[DISTINCT] wyr)*** - wywołanie Count(*) zwraca liczbę wszystkich wierszy uzyskanych w wyniku zapytania. Podanie argumentu w postaci wyrażenia spowoduje zwrócenie liczby wierszy, w których wybrane wyrażenie nie przyjęło wartości NULL,
- ***Max([DISTINCT] wyr)*** - zwraca największą wartość ze zbioru (opcjonalnie pozbawionego powtórzeń) pomijając NULL,
- ***Min([DISTINCT] wyr)*** - zwraca najmniejsza wartość ze zbioru (opcjonalnie pozbawionego powtórzeń) pomijając NULL,
- ***Stdev([DISTINCT] wyr)*** - wylicza odchylenie standardowe na zbiorze (opcjonalnie pozbawionym powtórzeń) pomijając NULL,
- ***Sum([DISTINCT] wyr)*** - wylicza sumę wartości ze zbioru (opcjonalnie pozbawionego powtórzeń) pomijając NULL,
- ***Variance(*|[DISTINCT] wyr)*** - Oblicza wariancje wartości ze zbioru (opcjonalnie pozbawionego powtórzeń) pomijając NULL,

Klauzula GROUP BY umożliwia podzielenie relacji na podzbiory - agregacje. Pojedynczy podzbiór/grupę stanowią wiersze, dla których kryterium grupowania ma identyczną wartość. Na liście wyrażeń klauzuli SELECT mogą wówczas stanąć jedynie te wyrażenia, które są przedmiotem działania klauzuli GROUP BY, oraz argumenty funkcji agregujących.

Przykłady



	ID_DZIALU	NAZWISKO
1	10	SKALSKI
2	10	KOWALSKA
3	10	KROL
4	20	WOJCIK
5	20	SZCZESNY
6	20	RYBAK
7	20	MONIUSZKO
8	20	WRZOSEK
9	20	LISIECKI
10	30	KUKULSKI
11	30	MALIK
12	30	GADULA
13	40	WISNIEWSKA
14	40	WIERZBICKI
15	40	MICHALSKI
16	40	LELIWA
17	50	NAWROCKI
18	50	PRUSINSKA
19	60	BRZOZKA
20	60	WOJCIK
21	60	MAZUR
22	60	BIELECKA
23	70	LESZCZYNSKI
24	70	KOWALCZYK
25	70	FIKUS
26	70	MALYSZ

	ID_DZIALU	Liczba pracownikow
1	10	3
2	20	6
3	30	3
4	40	4
5	50	2
6	60	4
7	70	4

Rys. 4 Wyniki zwrócone przez zapytania:

- bez grupowania: SELECT id_dzialu, nazwisko from pracownicy where data_zwol is NULL and id_dzialu is not NULL order by 1;
- z grupowaniem: SELECT **id_dzialu**, count(*) as "Liczba pracownikow" FROM pracownicy WHERE data_zwol is NULL and id_dzialu is not NULL **GROUP BY id_dzialu** ORDER BY 1;

	ROK	TRYB	Liczba studentow
1	1	STACJONARNY	595
2	1	NIESTACJONARNY	211
3	2	STACJONARNY	476
4	2	NIESTACJONARNY	182

Rys. 5 Fragment wyniku zwrócony przez zapytanie:
SELECT **rok**, **tryb**, count(*) as "Liczba studentow" FROM studenci GROUP BY **rok**, **tryb** ORDER BY 1, 3 DESC;

	TRYB	STOPIEN	KIERUNEK	ROK	Liczba grup dziekanskich	Liczba studentow	DU najstarszego studenta	DU najmodszego studenta	sredni wiek w dniach
1	NIESTACJONARNY	1	INFORMATYKA	1	2	61	83/06/22	98/11/09	8407
2	STACJONARNY	1	INFORMATYKA	1	8	233	83/05/19	98/12/29	8243
3	NIESTACJONARNY	1	INFORMATYKA	2	2	44	82/03/19	97/09/25	8618
4	STACJONARNY	1	INFORMATYKA	2	6	174	82/11/26	97/11/13	8582

Rys. 6 Fragment wyniku zwróconego przez zapytanie:
SELECT tryb, **stopien**, **kierunek**, **rok**, count(DISTINCT(gr_dziekan)) as "Liczba grup dziekanskich", count(*) as "Liczba studentow", min(data_urodzenia) "DU najstarszego studenta", max(data_urodzenia) as "DU najmodszego studenta", Round(avg(sysdate-data_urodzenia)) as "sredni wiek w dniach" **FROM** studenci **GROUP BY tryb**, **stopien**, **kierunek**, **rok** **ORDER BY** 2, 3, 4;

	STOPIEN	KIERUNEK	TRYB	ROK	GR_DZIEKAN	NVL(SPECJALNOSC,'BRAK')	Liczba roznych imion	Liczba studentow	DU najstarszego studenta
1	1	INFORMATYKA	STACJONARNY	3	1	INŻYNIERIA OPROGRAMOWANIA	23	25	82/12/22
2	1	INFORMATYKA	STACJONARNY	1	3	BRAK	24	27	83/05/19
3	1	MATEMATYKA	STACJONARNY	1	1	BRAK	24	25	88/10/19
4	1	MATEMATYKA	STACJONARNY	1	2	BRAK	24	26	94/03/23

Rys. 7 Fragment wyniku zwróconego przez zapytanie:
SELECT stopien, kierunek, tryb, rok, gr_dziekan, NVL(specjalnosc, 'BRAK'), count(DISTINCT(imiona)) as "Liczba roznych imion", count(*) as "Liczba studentow", min(data_urodzenia) "DU najstarszego studenta", max(data_urodzenia) as "DU najmodszego studenta", Round(avg(sysdate-data_urodzenia)) as "sredni wiek studenta w dniach"
FROM studenci
GROUP BY stopien, kierunek, tryb, rok, gr_dziekan, NVL(specjalnosc, 'BRAK')
HAVING count(DISTINCT(imiona)) BETWEEN 23 and 25 **ORDER BY** 1,2,4, 7 desc ;

	TRYB	STOPIEN	KIERUNEK	ROK	NAZWISKO	IMIONA	GR_DZIEKAN	SREDNIA
1	NIESTACJONARNY	1	INFORMATYKA	2	Janik-Szewczyk	Bernarda	2	5
2	STACJONARNY	1	INFORMATYKA	2	Wisniewski	Franciszek	2	5
3	STACJONARNY	1	MECHANIKA I BUDOWA MASZYN	2	Duda	Genowefa	3	4,98
4	STACJONARNY	1	MECHANIKA I BUDOWA MASZYN	3	Maj	Sławomir	1	4,98
5	NIESTACJONARNY	1	MECHANIKA I BUDOWA MASZYN	2	Wójtowicz-Marciniak	Kazimiera	1	4,96

Rys. 8 Fragment wyniku zwróconego przez zapytanie:

```
SELECT tryb, stopien,kierunek, rok, nazwisko, imiona, gr_dziekan, srednia FROM studenci
WHERE (tryb, stopien, kierunek,  srednia) IN
(SELECT tryb, stopien, kierunek, max(srednia) FROM studenci WHERE srednia IS NOT NULL GROUP BY kierunek, tryb, stopien)
ORDER BY 8 DESC;
```

	ROK	GR_DZIEKAN	SREDNIA_GRUPY
1	2	5	3,94
2	4	1	3,92
3	3	1	3,88
4	2	6	3,88

Rys. 9 Fragment wyniku zwróconego przez zapytanie:

```
SELECT * FROM
(SELECT rok, gr_dziekan, Round(AVG(srednia),2) as srednia_grupy FROM studenci
WHERE srednia IS NOT NULL AND tryb LIKE 'STACJONARNY' AND kierunek LIKE 'INFORMATYKA' AND stopien=1
GROUP BY rok, gr_dziekan)
ORDER BY srednia_grupy DESC;
```