Podstawy SQL

Kolejność używanych komend:

```
USE         --Zawężenie wyników do konkretnej bazy danych
SELECT      --Określenie wyświetlanych kolumn
FROM        --Określenie źródła danych (tabela)
WHERE       --Warunek, zawężenie wyników
GROUP BY    --Grupowanie rekordów
HAVING      --Warunek, zawężenie wyników
ORDER BY    --Sortowanie wyników
```

`USE baza_danych`	Instrukcja USE zapewnia nam wykonanie zapytania na odpowiedniej bazie danych.
`SELECT kolumna_1, kolumna_2...`	Instrukcja SELECT służy do pobrania danych z bazy. Określamy w niej interesujące nas kolumny oddzielone przecinkami (można też wprowadzić *, która zwraca nam wszystkie dostępne kolumny). Następnie korzystamy z klauzuli FROM. Dane zwracane są w kolejności odpowiadającej kolejności kolumn w zapytaniu.
`SELECT DISTINCT kolumna_1, kolumna_2...`	DISTINCT eliminuje duplikaty wierszy. Podając np. dwie kolumny po tej klauzuli, otrzymamy daną parę wartości tylko raz. Wyniki są automatycznie posortowane rosnąco.
`SELECT TOP n kolumna_1, kolumna_2...`	Klauzula TOP *n* zmniejsza nam ilość wyników. Gdzie *n* to liczba rekordów, które zostaną zwrócone.
`FROM nazwa_tabeli`	Za pomocą klauzuli FROM określamy tabelę, z której pobierane są dane (kolumny wymienione w SELECT pochodzą z tej tabeli).
`WHERE warunek`	Kluzula WHERE ogranicza wyniki do rekordów spełniających podany warunek / warunki.
`GROUP BY nazwa_kolumny`	Klauzula GROUP BY grupuje wyniki względem zawartości podanych kolumn. Serwer baz danych zwróci jeden wiersz dla każdej grupy wartości.
`HAVING warunek_z_funkcją_agregującą`	Kluzula HAVING podobnie jak WHERE ogranicza wyniki do rekordów spełniających podany warunek / warunki. Różnica polega na tym, że warunek wykorzystuje funkcję agregującą.
`ORDER BY nazwa_kolumny ASC` `ORDER BY nazwa_kolumny DESC`	Klauzula ORDER BY pozwal nam posortować wyniki po podanej kolumnie rosnąco (ASC) lub malejąco (DESC). Sortowanie rosnąco jest to wartość domyślna i nie jest konieczne podawanie wartości ASC.

OPERATORY PORÓWNAŃ – wykorzystywane najczęściej do budowania warunków zapytania (WHERE lub HAVING).

=	`WHERE nazwisko='Kowalski'`	Kolumna 'nazwisko' musi mieć dokładną wartość 'Kowalski'.
!= lub <>	`WHERE nazwisko!='Kowalski'` `WHERE nazwisko<>'Kowalski'`	Kolumna 'nazwisko' nie może mieć wartości 'Kowalski'.
>	`WHERE wiek > 20`	Kolumna 'wiek' musi posiadać wartość większą od 20.
>=	`WHERE wiek >= 20`	Kolumna 'wiek' musi posiadać wartość większą lub równą 20.
<	`WHERE wiek < 20`	Kolumna 'wiek' musi posiadać wartość mniejszą od 20.
<=	`WHERE wiek <= 20`	Kolumna 'wiek' musi posiadać wartość mniejszą lub równą 20.
BETWEEN *a* AND *b*	`WHERE wiek BETWEEN 20 AND 30`	Kolumna 'wiek' musi zawierać się w zakresie 20-30 (włącznie). *a* – wartość mniejsza, *b* – wartość większa
IN (a,b…)	`WHERE wiek IN (20,25,27)` `WHERE kraj IN ('Polska','Francja')`	Kolumna 'wiek' musi mieć jedną z wartości podanych w nawiasie.
IS NULL	`WHERE wiek IS NULL`	Kolumna 'wiek' nie może posiadać danej (w polu musi mieć *NULL*).
IS NOT NULL	`WHERE wiek IS NOT NULL`	Kolumna 'wiek' musi posiadać daną (w polu nie może mieć *NULL*).
LIKE	`WHERE nazwisko LIKE ...`	Kolumna 'nazwisko' musi być zgodna ze wzorcem (...). _ - zastępuje jeden znak % - zastępuje dowolny ciąg znaków
	`WHERE nazwisko LIKE 'Kowalski'`	Kolumna 'nazwisko' musi mieć wartość 'Kowalski' (analogicznie do znaku =).
	`WHERE nazwisko LIKE 'Kowalsk_'`	Kolumna 'nazwisko' musi zaczynać się od 'Kowalsk' i posiadać jeden dowolny znak na końcu (np. Kowalska, Kowalski).
	`WHERE nazwisko LIKE 'Kowal%'`	Kolumna 'nazwisko' musi zaczynać się od 'Kowal', końcówką może być dowolny ciąg znaków (np. Kowal, Kowaluk, Kowalski, Kowalczyk).
	`WHERE nazwisko LIKE '%owal%'`	Kolumna 'nazwisko' musi zawierać słowo 'owal', przed i po może być dowolny ciąg znaków.

NOT	`WHERE wiek NOT IN (20,25,27)`
	`WHERE nazwisko NOT LIKE ...`
	`WHERE wiek NOT BETWEEN 20 AND 30`
		Kolumna 'wiek' nie może mieć żadnej z wartości podanych w nawiasie.
		Kolumna 'nazwisko' nie może być zgodna ze wzorcem (...).
		Kolumna 'wiek' musi być spoza zakresu 20-30.

OPERATORY LOGICZNE – wykorzystywane najczęściej do łączenia warunków umieszczonych w klauzulach WHERE lub HAVING.
Należy pamiętać, że zawsze pierwszy pod uwagę brany jest 'AND', potem dopiero 'OR'.

AND	`WHERE nazwisko='Kowalski'` `AND wiek=25`	Muszą być spełnione jednocześnie oba warunki.
OR	`WHERE nazwisko='Kowalski' OR wiek=25`	Musi być spełniony jeden z podanych warunków.

OPERATORY MATEMATYCZNE – mogą być wykorzystywane zarówno po klauzuli SELECT, jak i w warunkach (WHERE, HAVING). Stosowane po klauzuli SELECT powodują utworzenie dodatkowej kolumny, do której trafi wynik 'działania'.

+
-
/
*
	1. `SELECT kolumna_1,`
	`(kolumna_1 + 10) AS new_column`
	
	2. `SELECT cena, ilość,`
	`(cena*ilość) AS zysk`
		1. Jako wynik zostanie wyświetlona kolumna_1 oraz dodatkowa kolumna o nazwie 'new_column', która będzie zawierała wartość kolumny_1 powiększoną o 10.
		2. Jako wynik zostanie wyświetlona kolumna 'cena', 'ilość' oraz dodatkowa kolumna o nazwie 'zysk', która będzie zawierała iloczyn tych dwóch kolumn.

ALIAS – nadanie innych nazw dla tabeli lub kolumny. Dzięki temu zapytanie jest bardziej czytelne.

`SELECT kolumna_1 AS [Imię],` `kolumna_2 AS [Nazwisko]`	Jako wynik zostanie wyświetlona kolumna_1 jako 'Imię', kolumna_2 jako 'Nazwisko'.

FUNKCJE AGREGUJĄCE – są to funkcje grupujące, dzięki którym możemy np. policzyć ilość wierszy spełniających określone kryteria. Funkcje te domyślnie nie eliminują powtarzających się wierszy.

UWAGA: Użycie funkcji agregujących wymusi na nas dodanie kolumn nieobjętych tą funkcją do klauzuli GROUP BY.

Funkcje COUNT, MIN, MAX można stosować do pól liczbowych i nieliczbowych.

Funkcje SUM, AVG można stosować tylko do pól liczbowych.

`COUNT(kolumna)`	`SELECT COUNT(EmployeeID)` `FROM Pracownicy`	Zwraca liczbę rekordów występujących w określonej kolumnie, ale nie zlicza *NULL*'i.
`SUM(kolumna)`	`SELECT SUM(Wynagrodzenie)` `FROM Pracownicy`	Zwrca sumę wartości występujących w określonej kolumnie.
`AVG(kolumna)`	`SELECT AVG(Wiek)` `FROM Pracownicy`	Zwraca średnią wartości występujących w określonej kolumnie.
`MIN(kolumna)`	`SELECT MIN(Cena)` `FROM Produkty`	Zwraca najmniejszą wartość występującą w określonej kolumnie.
`MAX(kolumna)`	`SELECT MAX(Cena)` `FROM Produkty`	Zwraca największą wartość występującą w określonej kolumnie.
`ROUND(kolumna,a)`	`SELECT ROUND(Cena,1)` `FROM Produkty`	Zwraca zaokrągloną liczbę do tylu miejsc po przecinku, ile wynosi drugi parametr '*a*'.

KONKATENACJA – łączenie ze sobą kilku ciągów tekstowych w jeden.

`1. SELECT (kolumna_1 + kolumna_2) AS [nazwa]`	1. Jako wynik zostanie wyświetlona kolumna o nazwie 'nazwa', której wartością będą połączone wartości kolumny_1 i kolumny_2.
`2. SELECT (Imię + ' ' + Nazwisko) AS [Full Name]`	2. Jako wynik zostanie wyświetlona kolumna 'Full Name', której wartością będzie wartość kolumny 'Imię' i kolumny 'Nazwisko' rozdzielone spacją (' ').
`3. SELECT ('Pan/Pani ' + Imię) AS [zwrot]`	3. Jako wynik zostanie wyświetlona kolumna 'zwrot', której wartością będzie tekst 'Pan/Pani ' połączony z wartością kolumny 'Imię'.

FORMATOWANIE DAT – zmiana formatu wyświetlania dat.

`SELECT CONVERT(varchar,kolumna_data,<code>)`	Zmienia format daty, znajdującej się w kolumnie *kolumna_data*, na format zgodny z podanym kodem (<code>). <code> - wartości podane w tabelce w skrypcie.
`SELECT FORMAT(kolumna_data,'format')`	Zmienia format daty, znajdującej się w kolumnie *kolumna_data*, na format zgodny z 'format' – np. 'dd-MM-yyyy'. Gdzie: dd – dzień MM – miesiąc yy lub yyy – rok hh – godzina mm – minuta ss – sekunda

FORMATOWANIE TEKSTU – wykorzystywane jest m.in. do formatowania tekstu znajdującego się w tabeli przed wykonaniem na nim zapytania. Pomaga to np. wyeliminować błędy związane z wielkością liter w wartościach tabeli.

`UPPER(kolumna)`	`SELECT kolumna_1`
	`FROM nazwa_tabeli`
	`WHERE UPPER(kolumna_1) = 'NAZWA'`
		Wyświetl rekordy kolumny_1 z tabeli 'nazwa_tabeli', których wartość, po zmianie wszystkich liter na wielkie, to 'NAZWA'.

`LOWER(kolumna)`	`SELECT kolumna_1`
	`FROM nazwa_tabeli`
	`WHERE LOWER(kolumna_1) = 'nazwa'`
		Wyświetl rekordy kolumny_1 z tabeli 'nazwa_tabeli', których wartość, po zmianie wszystkich liter na małe, to 'nazwa'.

PRZYCINANIE TEKSTU – wyświetlenie części tekstu z danej kolumny.

`SELECT SUBSTRING(kolumna,2, 4)` `FROM nazwa_tabeli`	Wyświetl część wartości kolumny 'kolumna' – od 2-go do 4-go znaku. Np. jeżeli wartość kolumny to *słowo* – wyświetlone będzie *łow*.

CASE – warunki – tworzy dodatkową kolumnę, która uzupełniana jest danymi wartościami zgodnie z opisanym warunkiem.

```
SELECT kolumna_1,
       CASE
         WHEN warunek
         THEN 'wartość, jeżeli prawda'
         ELSE 'wartość, jeżeli fałsz'
       END AS [alias]
FROM nazwa_tabeli
                          --ELSE nie jest konieczne. Zamiast wartości w polu znajdzie się wtedy NULL.
```

`SELECT kolumna_1,`
`  CASE`
`  WHEN kolumna_1 > 0`
`  THEN 'TAK'`
`  ELSE 'NIE'`
`  END AS [TakNie]`
`FROM nazwa_tabeli`
	Wyświetl kolumnę_1 z tabeli 'nazwa_tabeli' oraz dodatkową kolumnę 'TakNie'.
	Do kolumny '**TakNie**' wprowadź wartość '**TAK**', jeżeli wartość **kolumny_1 > 0**, natomiast wartość '**NIE**' wprowadź w pozostałych przypadkach.

DEKLAROWANIE ZMIENNYCH – utworzenie parametru danego typu o danej wartości. Jest to przydatne przy większych zapytaniach, gdy często odwołujemy się do tej samej wartości (tą wartość deklarujemy).

`DECLARE @nazwa_parametru typ_danej` `SET @nazwa_parametru = wartość`	Nazwa zmiennej musi być zawsze poprzedzona znakiem '@'. W miejsce 'typ_danej' wprowadzamy jeden z typów, np. INT, VARCHAR(n), DATE...

`DECLARE @kk INT`
`SET @kk = 24`
 	
`SELECT * FROM Employees`
`WHERE Wiek > @kk`
	Zadeklaruj zmienną o nazwie 'kk', której typ to INTEGER.
	Ustaw wartość zmiennej 'kk' jako równą 24.
	
	Wyświetl wszystkie rekordy z tabeli 'Employees',
	Których kolumna 'Wiek' ma wartość większą niż 24 (wartość zmiennej 'kk').

WIDOKI – tworzenie, modyfikowanie i usuwanie widoków .

`CREATE VIEW`	`CREATE VIEW nazwa_widoku AS` `SELECT ...`	Utwórz widok o nazwie 'nazwa_widoku'. Po SELECT wprowadzamy dalszy ciąg zapytania, którego wyniki chcemy zapisać jako widok.
`ALTER VIEW`	`ALTER VIEW istniejący_widok AS` `SELECT ...`	Zmień widok o nazwie 'istniejący_widok'. Po SELECT wprowadzamy dalszy ciąg zapytania, którego wyniki chcemy zapisać jako widok.
`DROP VIEW`	`DROP VIEW istniejący_widok`	Usuń widok o nazwie 'istniejący_widok'.

TABELE – dopisywanie, modyfikowanie oraz usuwanie danych z tabel.

`INSERT`	`INSERT INTO nazwa_tabeli (kolumny)` `VALUES (wartości)`	Dodaj do tabeli 'nazwa_tabeli' rekord. Wymienione kolumny zasil podanymi wartościami.

`UPDATE`	`UPDATE nazwa_tabeli`
	`SET kolumna_1 = wartość_1,`
	`  kolumna_2 = wartość_2`
	`WHERE warunek`
		W tabeli 'nazwa_tabeli' zmień zawartość pól kolumny_1 oraz kolumny_2 na wartości podane w zapytaniu, dla rekordów, które spełniają podany warunek.

`DELETE`	`DELETE FROM nazwa_tabeli` `WHERE warunek`	Z tabeli 'nazwa_tabeli' usuń rekordy, które spełniają podany warunek.

PODZAPYTANIA – odwołanie się w zapytaniu do wyniku innego zapytania.

`SELECT Country`
`FROM World`
`WHERE Area = (SELECT MAX(Area)`
`  FROM World)`
	Chcemy wyszukać nazwę kraju o największej powierzchni.
	Zapytanie: `SELECT MAX(Area) FROM World`
	zwróci nam wartość największej powierzchni.
	Wynik tego zapytania możemy wykorzystać do wyszukania kraju o tej powierzchni.
	Zamiast wprowadzać wartość liczbową wprowadzamy podzapytanie.

JOIN – łączenie jednej lub więcej tabel.

Tabela **BAJKA**

ID	Tytuł
1	Muminki
2	Smerfy
3	Kraina Lodu

Tabela **POSTAĆ**

ID	Bajka_ID	Imię
1	1	Mała Mi
2	3	Elsa
3	3	Olaf
4		Gucio
5		Shrek

INNER JOIN
JOIN

Zwraca nam wyłącznie te wiersze, dla których kolumny użyte do złączenia mają tę samą wartość.

```
SELECT *
FROM Bajka AS b
JOIN Postać AS p
ON b.ID=p.Bajka_ID
```

ID	Tytuł	ID	Bajka_ID	Imię
1	Muminki	1	1	Mała Mi
3	Kraina Lodu	2	3	Elsa
3	Kraina Lodu	3	3	Olaf

LEFT JOIN

Zwraca nam te wiersze, dla których kolumny użyte do złączenia mają tę samą wartość oraz wiersze z "lewej tabeli", dla których nie ma odpowiedników w prawej (wiersze te zawierają wartości NULL w kolumnach tabeli drugiej – „prawej").

```
SELECT *
FROM Bajka AS b
LEFT JOIN Postać AS p
ON b.ID=p.Bajka_ID
```

ID	Tytuł	ID	Bajka_ID	Imię
1	Muminki	1	1	Mała Mi
2	Smerfy	*NULL*	*NULL*	*NULL*
3	Kraina Lodu	2	3	Elsa
3	Kraina Lodu	3	3	Olaf

RIGHT JOIN

Zwraca nam te wiersze, dla których kolumny użyte do złączenia mają tę samą wartość oraz wiersze z "prawej tabeli", dla których nie ma odpowiedników w lewej (wiersze te zawierają wartości NULL w kolumnach tabeli drugiej – „lewej").

```
SELECT *
FROM Bajka AS b
RIGHT JOIN Postać AS p
ON b.ID=p.Bajka_ID
```

ID	Tytuł	ID	Bajka_ID	Imię
1	Muminki	1	1	Mała Mi
3	Kraina Lodu	2	3	Elsa
3	Kraina Lodu	3	3	Olaf
NULL	*NULL*	4		Gucio
NULL	*NULL*	5		Shrek

FULL JOIN

Zwraca nam te wiersze, dla których kolumny użyte do złączenia mają tę samą wartość, wiersze z "lewej tabeli", dla których nie ma odpowiedników w prawej oraz wiersze z "prawej tabeli", dla których nie ma odpowiedników w lewej.

```
SELECT *
FROM Bajka AS b
FULL JOIN Postać AS p
ON b.ID=p.Bajka_ID
```

ID	Tytuł	ID	Bajka_ID	Imię
1	Muminki	1	1	Mała Mi
2	Smerfy	*NULL*	*NULL*	*NULL*
3	Kraina Lodu	2	3	Elsa
3	Kraina Lodu	3	3	Olaf
NULL	*NULL*	4		Gucio
NULL	*NULL*	5		Shrek