Spis treści

[AND 2](#_Toc105089628)

[OR 2](#_Toc105089629)

[BETWEEN 2](#_Toc105089630)

[IN 3](#_Toc105089631)

[ORDER BY 3](#_Toc105089632)

[WHERE 3](#_Toc105089633)

[LIKE 4](#_Toc105089634)

[LENGTH 4](#_Toc105089635)

[JOIN && RIGHT JOIN && LEFT JOIN && FULL JOIN 5](#_Toc105089636)

[UNION 5](#_Toc105089637)

[INTERSECT 6](#_Toc105089638)

[MINUS 7](#_Toc105089639)

[MIN() 7](#_Toc105089640)

[MAX() 7](#_Toc105089641)

[COUNT() 7](#_Toc105089642)

[AVG() 8](#_Toc105089643)

[SUM() 8](#_Toc105089644)

[GROUP BY 8](#_Toc105089645)

[HAVING 8](#_Toc105089646)

[ANY 8](#_Toc105089647)

[ALL 9](#_Toc105089648)

[1 10](#_Toc105089649)

[2 10](#_Toc105089650)

[3 10](#_Toc105089651)

[**4** 11](#_Toc105089652)

[INSERT INTO (ALL) 11](#_Toc105089653)

[UPDATE 12](#_Toc105089654)

[CREATE TABLE 12](#_Toc105089655)

[ALTER TABLE 12](#_Toc105089656)

[PRIMARY KEY 13](#_Toc105089657)

CORE

# AND

SELECT `nazwa\_kolumny`  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny1`='wartość1'  
AND `nazwa\_kolumny2`='wartość2'

SELECT \*  
FROM `osoby`  
WHERE `imię`='Michał'  
AND `nazwisko`='Nowak'

# OR

SELECT \*  
FROM `osoby`  
WHERE `imię`='Paweł'  
OR `nazwisko`='Nowak'

SELECT \*  
FROM `osoby`  
WHERE `imię`='Paweł'  
OR `nazwisko`='Nowak'

# BETWEEN

SELECT `nazwa\_kolumny`  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny`  
BETWEEN 'wartość1' AND 'wartość2'

SELECT `imię`,`nazwisko`  
FROM `osoby` WHERE `wiek`  
BETWEEN 3 AND 13

# IN

SELECT `nazwa\_kolumny` FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny`  
IN ('wartość1','wartość2')

SELECT `imię`,`nazwisko`  
FROM `osoby`  
WHERE `nazwisko`  
IN ('Kowalski','Nowak')

# ORDER BY

*Klauzula* ***ORDER BY*** *(opcjonalna) sortuje zwracane w zapytaniu rekordy rosnąco (****ASC*** *– domyślne, można opuścić) lub malejąco (****DESC****) względem kolumn (jednej lub kilku) wyspecyfikowanych po klauzuli. W zapytaniu może pojawić się tylko raz, na końcu.*

SELECT `nazwa\_kolumny`  
FROM `nazwa\_tabeli`  
[WHERE `nazwa\_kolumny`]  
ORDER BY 'nazwa\_kolumny' [ASC,DESC]  
[, 'nazwa\_kolumny2' [ASC,DESC] ]

SELECT \*  
FROM `osoby`  
ORDER BY `imie`,`nazwisko`

# WHERE

*Klauzula* ***WHERE*** *(opcjonalna) ograniczająca zwracane rekordy do tych tylko, które dla podanego po* ***WHERE*** *warunku logicznego zwracają* ***TRUE*** *(odrzucane są te, które zwracają* ***FALSE*** *lub* ***NULL****). Warunek logiczny może być zbudowany z wielu zdań logicznych, połączonych negacją* ***NOT****, koniunkcją* ***AND*** *lub alternatywą* ***OR.***

SELECT `nazwa\_kolumny`  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny` operator 'wartosc'

SELECT `imie`,`nazwisko`  
FROM `osoby`  
WHERE `wiek` > 6

*Jako operator można użyć : = , !=,< , > , <= , >= , BETWEEN , LIKE ,IN*

# LIKE

*Predykat* ***LIKE*** *oraz wildcards, czyli znaki uniwersalne: ‘\_’ (zastępujący jeden dowolny znak) i ‘%’ (zastępujący dowolny ciąg znaków).*

SELECT `nazwa\_kolumny`  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny`  
LIKE wzorzec

SELECT `imię`,`nazwisko`   
    FROM `osoby`  
    WHERE `nazwisko`  
    LIKE '%k'

%X -KOŃCZY SIĘ

X%-ZACZYNA SIĘ

%XXX%-MIĘDZY POCZ. A KOŃ.

# LENGTH

***LENGTH*** *(Oracle) i* ***LEN*** *(Ms SQL Server)*.

SELECT LENGTH(`nazwa\_kolumny`)  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny`

SELECT `nazwisko`,LENGTH(`nazwisko`)  
FROM `osoby`  
GROUP BY `nazwisko`

--WHERE (job = 'MANAGER' OR job = 'SALESMAN') AND sal > 1500;

ŁĄCZENIE TABEL

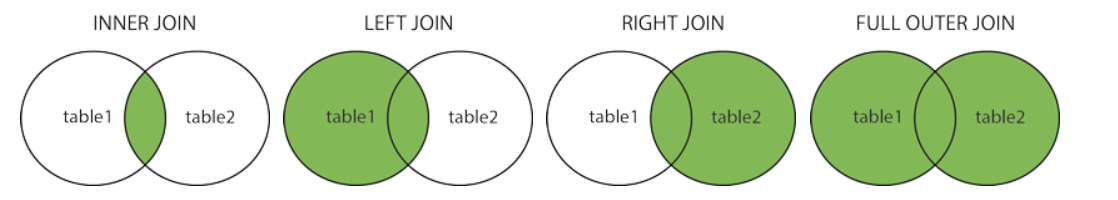
# JOIN && RIGHT JOIN && LEFT JOIN && FULL JOIN

INNER JOIN zwraca wszystkie wiersze jednej tabeli nawet jeśli nie wszystkie są dopasowane z drugą tabelą. Wyróżnia się

LEFT JOIN - zwraca wszystkie wiersze lewej tabeli nawet jeśli nie są dopasowane z prawą tabelą

RIGHT JOIN - zwraca wszystkie wiersze prawej tabeli nawet jeśli nie są dopasowane z lewą tabelą

FULL JOIN – WSZYSTKO



SELECT `nazwa\_kolumny1`,`nazwa\_kolumny2`  
FROM `nazwa\_tabeli1`  
INNER JOIN `nazwa\_kolumny2`  
ON `nazwa\_tabeli1`.`nazwa\_kolumny1`=`nazwa\_tabeli2`.`nazwa\_kolumny2`

SELECT \* FROM `osoby`  
INNER JOIN `zamowienia`  
ON `zamowienia`.`id\_osoby`=`osoby`.`id`

# UNION

SELECT job

FROM Emp

WHERE deptno = 10

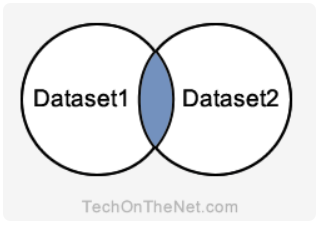
INTERSECT

SELECT job

FROM Emp

WHERE deptno = 30;

# INTERSECT

****

SELECT expression1, expression2, ... expression\_n

FROM tables

[WHERE conditions]

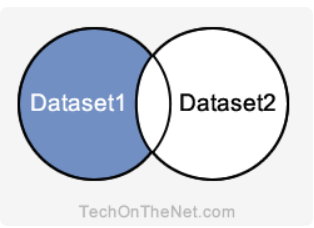
INTERSECT

SELECT expression1, expression2, ... expression\_n

FROM tables

[WHERE conditions];

# MINUS

****

SELECT supplier\_id, supplier\_name

FROM suppliers

WHERE supplier\_id > 2000

MINUS

SELECT company\_id, company\_name

FROM companies

WHERE company\_id > 1000

GRUPOWANIE REKORDÓW (FUNKCJI AGREGUJĄCE)

# MIN()

SELECT MIN(`nazwa\_kolumny`)  
FROM `nazwa\_tabeli`

# MAX()

SELECT MAX(`nazwa\_kolumny`)  
FROM `nazwa\_tabeli`

# COUNT()

SELECT COUNT([DISTINCT lub 1] `nazwa\_kolumny`)  
FROM `nazwa\_tabeli`

# AVG()

SELECT AVG(`nazwa\_kolumny`)

FROM `nazwa\_tabeli`

# SUM()

SELECT SUM(`nazwa\_kolumny`)  
FROM `nazwa\_tabeli`

# GROUP BY

SELECT `nazwa\_kolumny`, funkcja\_agregująca(`nazwa\_kolumny`)  
FROM `nazwa\_tabeli`  
WHERE `nazwa\_kolumny` operator 'wartość'  
GROUP BY `nazwa\_kolumny`

SELECT `imię`,`nazwisko`,sum(`wartosc\_zamowienia`)  
FROM `zamowienia`  
GROUP BY `imię`,`nazwisko`

# 

# HAVING

SELECT `nazwa\_kolumny1`,SUM(`nazwa\_kolumny2`)  
FROM `nazwa\_tabeli`  
GROUP BY `nazwa\_kolumn1`  
HAVING funkcja\_agregująca operator 'wartość'

SELECT `imie`,`nazwisko`,SUM(`wartosc\_zamowienia`)  
FROM `zamowienia`  
GROUP BY `imie`,`nazwisko`  
HAVING SUM(`zamowienia`)>200

CZ10

# ANY

SELECT \*

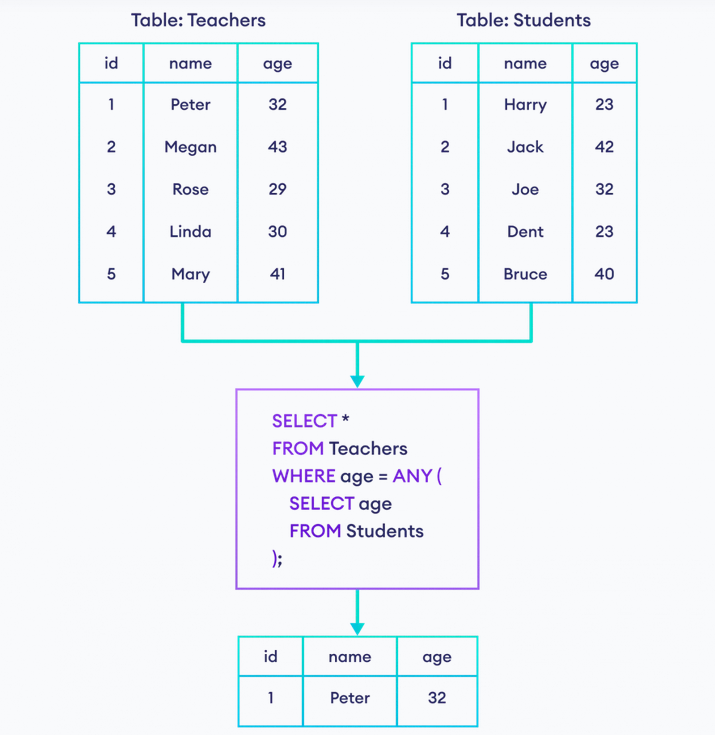
FROM Teachers

WHERE age = ANY (

SELECT age

FROM Students

);



# ALL

SELECT \*

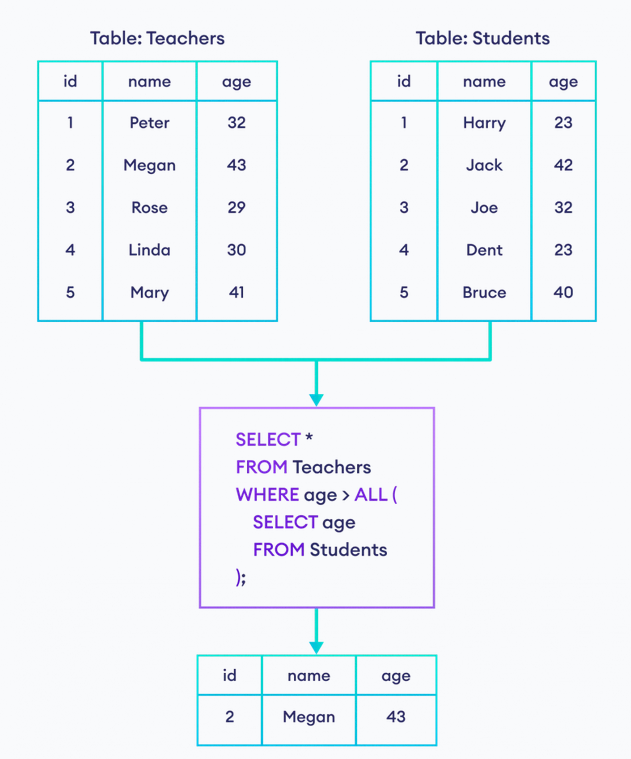
FROM Teachers

WHERE age > ALL (

SELECT age

FROM Students

);



CZ11

# 1

SELECT job, empno, ename, sal

FROM Emp e1

WHERE sal = (SELECT MAX(sal)

FROM Emp e2

WHERE e1.job = e2.job);

SELECT \* FROM Emp;

# 2

SELECT dname, loc

FROM Dept d

WHERE NOT EXISTS (SELECT 1

FROM Emp e

WHERE d.deptno = e.deptno);

# 3

Podaj nazwisko, pensję i nazwę departamentu pracowników, których płaca przekracza średnią ich grup zarobkowych.

SELECT ename, sal, dname

FROM Emp e1

JOIN Dept d1

ON e1.deptno = d1.deptno

JOIN Salgrade s1

ON e1.sal BETWEEN s1.losal AND s1.hisal

WHERE sal > (SELECT Avg(sal)

FROM Emp e2

JOIN Salgrade s2

ON s1.grade = s2.grade)

ORDER BY s1.Grade Desc;

select \* from salgrade;

# **4**

Znajdź nazwiska, płace i grupy zarobkowe pracowników zarabiających minimalną pensję w ich grupach zarobkowych. Wynikowe rekordy uporządkuj według malejących grup zarobkowych.

SELECT ename, sal, grade

FROM Emp e1

JOIN Salgrade s1

ON sal BETWEEN s1.losal AND s1.hisal

WHERE Sal = (SELECT MIN(sal)

FROM Emp e2

JOIN Salgrade s2

ON sal BETWEEN s2.losal AND s2.hisal

WHERE s1.grade = s2.grade)

ORDER BY grade DESC;

ADDING TO TAB

# INSERT INTO (ALL)

***INSERT INTO*** *pozwala na wstawianie nowych wartości do tabeli*

INSERT INTO `nazwa\_tabeli`  
VALUES ('wartość','wartość2')

INSERT INTO `nazwa\_tabeli` (`nazwa\_kolumny1`,`nazwa\_kolumny2`)  
VALUES ('wartość','wartość2')

# UPDATE

***UPDATE*** *pozwala na zmianę, aktualizację wartości kolumny w tabeli*

UPDATE `nazwa\_tabeli`  
SET `nazwa\_kolumny`='wartość'  
[WHERE `nazwa\_kolumny` operator 'wartość']

UPDATE `osoby`  
SET `wiek`=30  
WHERE `imie`='Jan' AND `nazwisko`='Kowalski'

# CREATE TABLE

***CREATE TABLE*** *tworzy tablice*

CREATE TABLE `nazwa\_tablicy`  
{  
`nazwa\_kolumny1` typ\_danych  
`nazwa\_kolumny2` typ\_danych  
...  
}

CREATE TABLE `osoby`  
{  
`imie` text  
`nazwisko` text  
`PESEL` int  
}

# ALTER TABLE

*░The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.*

*░The ALTER TABLE statement is also used to add and drop various constraints on an existing table.*

ALTER TABLE *table\_name*  
ADD *column\_name datatype*;

MODIFY COLUMN column\_name datatype;

MODIFY column\_name datatype;

DROP COLUMN column\_name;

ALTER COLUMN column\_name datatype;

# PRIMARY KEY

CREATE TABLE `nazwa\_tabeli`  
{  
`nazwa\_kolumny1` typ\_danych,  
`nazwa\_kolumny2` typ\_danych,  
PRIMARY KEY (`nazwa\_kolumny1`)  
}

*--NA USTALENIA*

ALTER TABLE `nazwa\_tabeli`  
ADD PRIMARY KEY (`nazwa\_kolumny`)

*--NA USUWANIE*

ALTER TABLE `nazwa\_tabeli`  
DROP PRIMARY KEY

ADDITIONAL

***▓ SELECT***

(sal+250)\*12, - działanie z kolumnami

'Pracownik ' || empno || ' ' || ename || ' pracuje w dziale nr ' || deptno "Informacje o pracownikach" – zmienanie nazwy kolumn.

***▓ NVL*** *(Oracle) i* ***ISNULL*** *(Ms SQL Server) zwraca wartość pierwszego argumentu, jeżeli jest on znaczący albo wartość drugiego argumentu, jeżeli pierwszy jest NULL.*

SELECT ename, sal\*12 + NVL(comm, 0) FROM EMP;

*▓ Powtarzające się rekordy eliminuje opcjonalne* ***DISTINCT*** *w klauzuli* ***SELECT***

SELECT DISTINCT deptno FROM EMP;

*▓ Użycie nawiasów zmienia normalną hierarchię operatorów logicznych* ***NOT****,* ***AND****,* ***OR***

*▓* Wypisz nazwisko, numer departamentu i datę zatrudnienia tych pracowników, którzy zostali zatrudnieni w 1982 r.

*Jako alternatywne rozwiązanie wykorzystaj operator LIKE (nie jest to dobre rozwiązanie!). Lepiej jest skorzystać ze sprawdzenia zakresu dat lub funkcji zwracającej rok. Są to funkcje* ***YEAR****(data) w MS SQL Server i* ***EXTRACT****(Year* ***FROM*** *data) w ORACLE.*

SELECT ename, deptno, hiredate

FROM Emp

WHERE EXTRACT(Year FROM hiredate) = 1982;