

Funkcje agregujące, grupowanie danych w phpMyAdmin.

Łączenie tabel w phpMyAdmin.

Funkcje agregujące, grupowanie danych w phpMyAdmin.

Funkcje agregujące, to takie funkcje, które zwracają jedną wartość wyliczoną na podstawie wielu wierszy. Wszystkie funkcje grupowe ignorują wiersze zawierające wartość **null** w kolumnie, na której działają.

Opcjonalnie do każdej funkcji można dodać modyfikator **DISTINCT** co będzie skutkowało przeliczeniem nie uwzględniającym wartości, które się powtarzają.

Niektóre funkcje agregujące

Nazwa	Opis
AVG(kolumna)	Oblicza średnią arytmetyczną w danej kolumnie
COUNT(kolumna)	Oblicza liczbę niezerowych wartości w kolumnie
COUNT(DISTINCT kolumna)	Oblicza liczbę różnych wartości w kolumnie
COUNT(*)	Oblicza liczbę wierszy w tabeli
MIN(kolumna) lub MAX(kolumna)	Oblicza najmniejszą lub największą wartość w kolumnie
SUM(kolumna)	Oblicza sumę wartości w kolumnie

W przypadku funkcji agregujących nazwy kolumn również tracą swoją nazwę, dlatego i tu najlepiej użyć aliasów, by później wiedzieć co dana kolumna przedstawia.

Funkcji agregujących możemy używać także kilku na raz w jednym zapytaniu.

Grupowanie

Grupowanie polega na podzieleniu zbioru wierszy na grupy, które mają pewną wspólną cechę. Grupowania dokonuje się w celu zastosowania funkcji agregujących nie w stosunku do całego zbioru wierszy, ale do poszczególnych grup wierszy. W celu zgrupowania rekordów należy dodać nową klauzulę **GROUP BY** wraz z wyspecyfikowaniem kolumny lub wyrażenia, według którego mają być pogrupowane wiersze. Na liście klauzuli **SELECT** mogą się znaleźć tylko kolumny i wyrażenia, według których zapytanie jest grupowane, oraz wywołania funkcji agregujących. Przed grupowaniem można wcześniej wyfiltrować wiersze, które nas interesują, za pomocą klauzuli **WHERE**.

Operator HAVING

Przed dokonaniem grupowania można zastosować klauzulę **WHERE**, która wybierze tylko część wierszy z tabeli. Klauzula **WHERE** wykonuje się przed grupowaniem, a zatem nie można w tej klauzuli sprecyzować warunku zawierającego funkcje grupowe. Aby taki warunek zawrzeć w zapytaniu należy zastosować dodatkową klauzulę **HAVING** wraz z odpowiednim warunkiem. Jest ona odpowiednikiem klauzuli **WHERE**, tylko, że wykonuje się ona po procesie grupowania.

Przykłady:

- Średnia cena wszystkich produktów, zaokrąglona do 2 miejsc po przecinku:

```
SELECT round(AVG(cena),2) as srednia_cena FROM produkt
```

- Średnia, minimalna i maksymalna cena wszystkich produktów

```
SELECT round(AVG(cena),2) as srednia_cena, min(cena) as minimalna_cena,
max(cena) as maksymalna_cena FROM produkt;
```

srednia_cena	minimalna_cena	maksymalna_cena
711.22	2.7	3700

- Liczba wszystkich produktów:

```
SELECT count(id_produkt) as liczba_produktow FROM produkt
```

- Łączna cena wszystkich produktów:

```
SELECT SUM(cena) as laczna_cena FROM produkt;
```

- Liczba klientów z miast, które się nie powtarzają:

```
SELECT count(DISTINCT miasto) AS liczba_klientow FROM klienci;
```

- Liczba klientów z miast, które się powtarzają:

```
SELECT count(miasto) AS liczba_klientow FROM klienci;
```

- Liczba klientów z danego miasta, posortowane od największej liczby klientów:

```
SELECT count(klient_id) AS liczba_klientow,miasto
FROM klienci
GROUP BY miasto
ORDER BY liczba_klientow DESC;
```

- Wyświetlenie tylko tych miast, które mają więcej niż 2 klientów:

```
SELECT count(klient_id) AS liczba_klientow,miasto
FROM klienci
GROUP BY miasto
HAVING count(klient_id) >2
ORDER BY liczba_klientow DESC;
```

- Wyświetlenie tylko tych miast, które mają więcej niż 1 klienta i nie pochodzą z Rzeszowa

```
SELECT count(klient_id) AS liczba_klientow,miasto
FROM klienci
WHERE miasto<>'Rzeszow'
GROUP BY miasto
HAVING count(klient_id) >=2
ORDER BY liczba_klientow DESC;
```

Łączenie tabel w phpMyAdmin

Złączenia tabel dokonujemy, jeśli chcemy połączyć i odczytać dane z więcej niż jednej tabeli. Przed utworzeniem zapytania łączącego tabele powinniśmy sprawdzić:

- czy w łączonych tabelach są unikalne rekordy – jeśli w tabelach są duplikaty (ten sam rekord występuje więcej niż raz), złączenie tabel spowoduje duplikację rekordów
- według których kolumn chcemy połączyć tabele
- czy złączenie tabel nie spowoduje problemów z wydajnością bazy – im więcej złączeń tym dłuższy czas realizacji zapytania przez serwer

Rodzaje złączeń w MS SQL Server

- złączenie wewnętrzne – INNER JOIN (JOIN)
- złączenie zewnętrzne lewostronne – LEFT OUTER JOIN (LEFT JOIN)
- złączenie zewnętrzne prawostronne- RIGHT OUTER JOIN (RIGHT JOIN)
- złączenie zewnętrzne obustronne- FULL OUTER JOIN (FULL JOIN)

Złączenie wewnętrzne JOIN

W wyniku złączenia wewnętrznego (INNER JOIN, JOIN) otrzymujemy tylko te rekordy, dla których warunki złączenia będą spełnione (wynik będzie TRUE).

Poszczególne kroki łączenia tabel w JOIN:

1. wykonanie iloczynu kartezjańskiego obu tabel – połączenie każdego wiersza z tabeli A z każdym wierszem z tabeli B
2. sprawdzenie dla każdej pary wierszy warunku określonego w operatorze ON.
3. odrzucenie wszystkich wierszy niespełniających warunków określonych w operatorze ON. W zbiorze wynikowym pozostają tylko te wiersze, dla których warunek określony w operatorze ON jest spełniony (przyjmuje wartość TRUE).

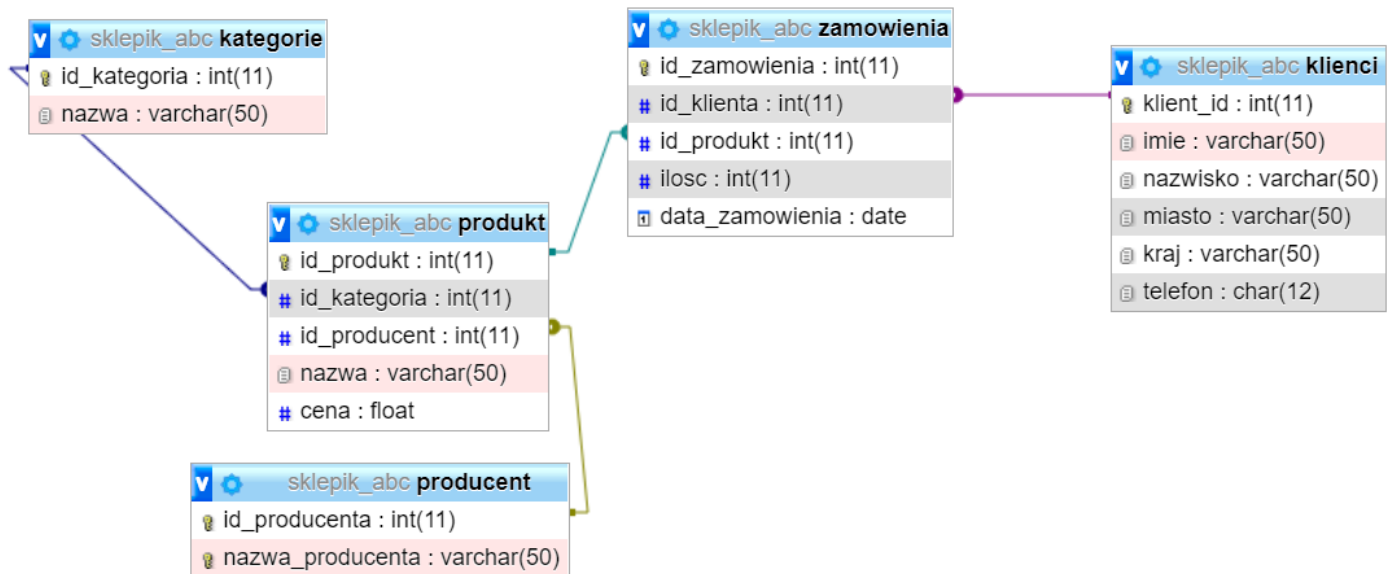
JOIN jest złączeniem symetrycznym – nie ma znaczenia w jakiej kolejności łączymy tabele A z B czy odwrotnie. Nie ma też znaczenia kolejność, w jakiej wypiszemy warunki złączenia w słowie kluczowym ON.

W przypadku złączenia JOIN, warunki złączenia możemy wymienić w klauzuli ON lub w klauzuli WHERE. Łatwiejsze jest analizowanie zapytania, jeśli złączenie realizujemy przy wykorzystaniu klauzuli JOIN, gdyż:

- w klauzuli JOIN mamy wymienione wszystkie łączone tabele i warunki ich złączenia
- w klauzuli WHERE mamy jedynie warunki ograniczające liczbę rekordów (filtrujące rekordy) i nie musimy analizować, które warunki dotyczą złączenia, a które filtrowania rekordów.

Wyjaśnienia łączenia tabel na bazie sklepik_abc

sklepik_abc

**Przykład:**

1. Wyświetlenie nazwy i ceny produktu oraz nazwy kategorii.

```

SELECT produkt.nazwa, produkt.cena, kategorie.nazwa
FROM produkt JOIN kategorie
ON kategorie.id_kategoria=produkt.id_kategoria;
  
```

nazwa	cena	nazwa
Mikser	300	AGD
robot sprzątający	1700	AGD
lalka zuzia	25	zabawki
marchewka 300g	4.5	spożywcze
pralka	1700	AGD
suszarka	2200	AGD
sandały dziecięce r.22	35	obuwie
sandały dziecięce r. 24	42	obuwie

2. Wyświetl dane klienta, który zapłacił najwięcej za zamówienie i cenę jaką zapłacił.

```

SELECT klienci.imie, klienci.nazwisko, (zamowienia.ilosc*produkt.cena) AS
kwota
FROM klienci JOIN zamowienia
ON zamowienia.id_klienta=klienci.klient_id
JOIN produkt
ON produkt.id_produkt=zamowienia.id_produkt
ORDER BY kwota DESC LIMIT 1
  
```

imie	nazwisko	kwota
Ewelina	Nowak	4400

Złączenie zewnętrzne lewostronne (LEFT JOIN, LEFT OUTER JOIN)

W złączeniu typu LEFT JOIN łączone są najpierw wewnętrznie dwa zbiory (realizowany jest INNER JOIN). Następnie do wyniku zapytania dodawane są wszystkie niedopasowane elementy tabeli po PRAWEJ stronie operatora JOIN.

W wyniku złączenia LEFT JOIN otrzymamy:

- wszystkie rekordy z tabeli po LEWEJ stronie operatora JOIN
- dopasowane rekordy z tabeli po PRAWEJ stronie operatora JOIN
- niedopasowane rekordy z tabeli po PRAWEJ stronie operatora JOIN

Złączenie zewnętrzne prawostronne (RIGHT JOIN, RIGHT OUTER JOIN)¹

Tak samo, jak w przypadku złączenia LEFT JOIN, w złączeniu RIGHT JOIN łączone są najpierw wewnętrznie dwa zbiory (INNER JOIN). Następnie do wyniku zapytania dodawane są wszystkie niedopasowane elementy tabeli po LEWEJ stronie operatora JOIN.

W wyniku złączenia RIGHT JOIN otrzymamy:

- wszystkie rekordy z tabeli po PRAWEJ stronie operatora JOIN
- dopasowane rekordy z tabeli po LEWEJ stronie operatora JOIN
- niedopasowane rekordy z tabeli po LEWEJ stronie operatora JOIN

Przykłady:

1. Wyświetlenie klientów, którzy nie składali zamówień:

```
SELECT klienci.imie, klienci.nazwisko, zamowienia.id_klienta
FROM klienci LEFT JOIN zamowienia
ON zamowienia.id_klienta=klienci.klient_id
WHERE zamowienia.id_klienta IS NULL;
```

Można stosować aliasami

```
SELECT k.imie, k.nazwisko, z.id_klienta
FROM klienci AS k LEFT JOIN zamowienia AS z
ON z.id_klienta=k.klient_id
WHERE z.id_klienta IS NULL;
```

¹ [MS SQL złączenia tabel na podstawie praktycznych przykładów \(avendi.edu.pl\)](https://avendi.edu.pl)

imie	nazwisko	id_klienta
Piotr	Czechosłowski	NULL
Anna	Kowalska	NULL
Paweł	Zygmunt	NULL
Aniela	Makówka	NULL

2. Wyświetlenie produktów, które nie były zamówione:

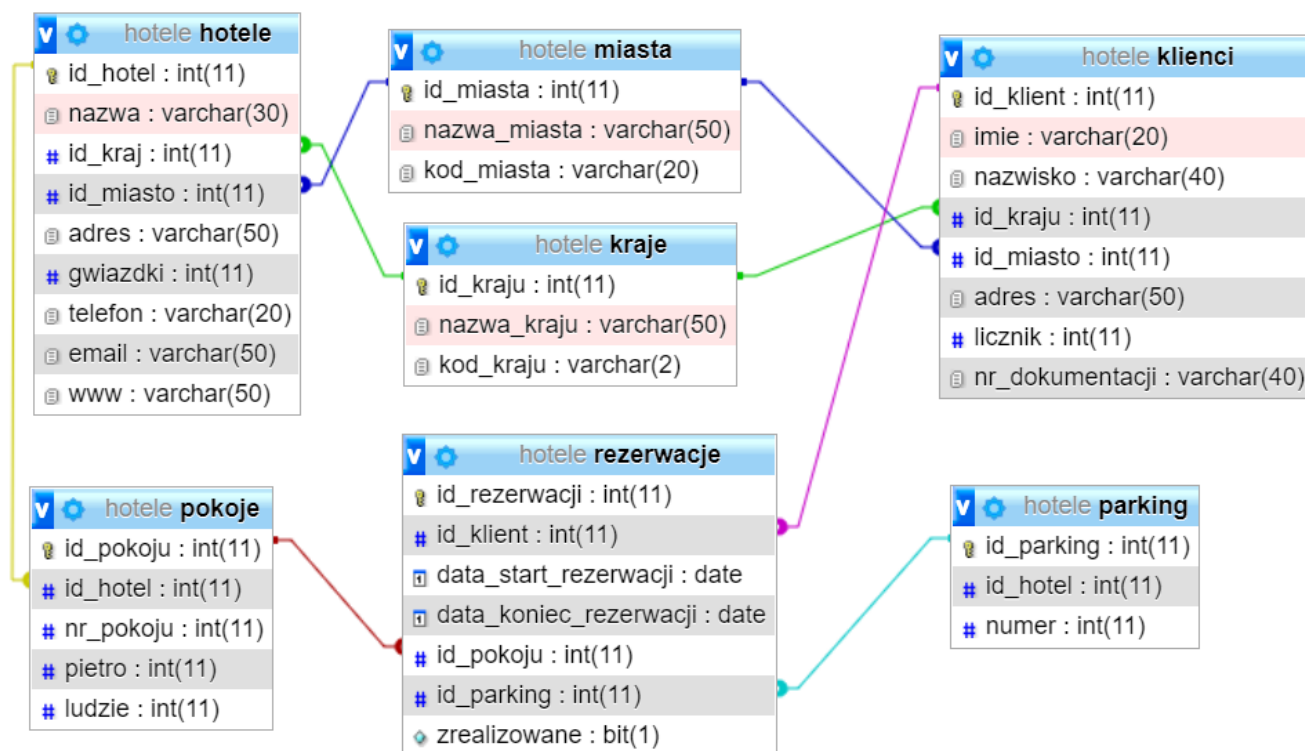
Wstaw do tabeli produkty, dwa dowolne produkty, a następnie wykonaj poniższe zapytanie.

```
SELECT produkt.nazwa, zamowienia.id_produkt
FROM produkt LEFT JOIN zamowienia
ON zamowienia.id_produkt=produkt.id_produkt
WHERE zamowienia.id_produkt IS NULL
```

nazwa	id_produkt
piłka zmyłka	NULL
frytkownica	NULL

Zadanie w domu lub samodzielnie na lekcji.

Zaimportuj bazę, a następnie wykonaj kwerendy. Część z nich możesz wykonać w postaci widoków.

Przykładowa baza danych – Hotele**schemat_hotele****Zapytania:**

- Wyświetl listę hoteli które mają między 3 a 5 gwiazdek.
- Wyświetl minimalną, maksymalną i średnią liczbę gwiazdek w hotelach.
- Wyświetl nazwy hoteli według liczby rezerwacji, od najwyższej liczby rezerwacji do najniższej.
- Czy jest jakiś klient, który nic nie rezerwował?
- Wyświetl klienta, który najczęściej dokonywał rezerwacji.
- Wyświetl klienta, który zarezerwował pokój na najdłuższą liczbę dni.
- Z którego miasta pochodzi najwięcej klientów, wyświetl miasto i liczbę wystąpień.
- Wyświetl 5 najrzadziej rezerwowanych hoteli.
- W jakim kraju jest najwięcej hoteli, podaj kraj i liczbę.
- Wyświetl ilu klientów jest z poszczególnych krajów. Posortuj po liczbie wystąpień od najwyższej do najniższej.
- Który pokój był najdłużej zarezerwowany?
- Wyświetl imię, nazwisko, miasto, kraj klienta, datę rezerwacji końca i początku, i nazwę hotelu w którym zarezerwował pokój.